VEJ TIL DRIFT

En teknisk redegørelse

Dato: 23-01-2026

Status: Final

Version: 1.1

Formål

Dette dokument er en større teknisk redegørelse for de løsninger udviklet og afprøvet i projektet. Dokumentet har til formål at besvare følgende spørgsmål:

- Hvad har vi bygget?
- Hvor godt virkede det?
- Hvordan kan man forbedre på løsningerne?
- Hvad skal man være opmærksom på i vejen mod drift?

Resume

Alle 3 Use cases har deres tekniske løsning beskrevet, evalueringsresultater gennemgået og videreudviklings- og idriftsættelsesanbefalinger fremlagt.

Den tekniske løsning, herunder afprøvningsklienter og data science komponenter er beskrevet for hver af projektets 3 Use cases (UC):

- UC1: Sygeplejefaglig udredning
- UC2: Journalopsummering
- UC3: Indtaling af besøgsbeskrivelser

For et overordnet teknisk overblik over hver Use case se for UC1 figur 1, for UC2 figur 7 og for UC3 figur 10.

Projektets primære metode for at vurdere kvaliteten af vores løsninger var indsamling af feedback fra fagligt personale i forbindelse med interne- (i de 3 projektkommuner) og eksterne (i 11 afprøvningskommuner) afprøvninger. For alle Use cases er afprøvningerne understøttet af en afprøvningsklient. Data indsamlet fra afprøvningerne er opgjort for hver Use case, centrale resultater for UC1 fremgår af tabel 4, for UC2 af tabel 9a og 9b, og for UC3 af figur 12.

I forlængelse af projektets resultater gennemgås for hver Use case en række anbefalinger til videreudvikling og idriftsættelse af løsningerne.

Fokus for videreudvikling af UC1 er etablering af et evalueringssæt, iterativ værdiskabelse og komponentforbedringer. For UC2 er fokus udvikling af løsningen med brug af ægte data, da syntetiske borgerjournaler i projektet ikke korrekt reflekterede kompleksiteten og variationen af den ægte vare. For UC3 er fokus på begrænsning af hallucinationer, brug af mindre modeller og etablering af kvalitetstjek.

Indholdsfortegnelse

Formål 2
Resume 2
1. Introduktion 6
2. Sygeplejefaglig udredning (UC1) 7
2.1 Formål og beskrivelse 7
2.2 Teknisk løsning 7
Afprøvningsklient 8
Talegenkendelse 12
Kategorisering 14
Opsummering – Promptstruktur ("Hvordan har vi opsat vores prompting") 17
Analyser 21
2.3 Evaluering 24
Evaluering under udvikling 25
Evaluering af afprøvninger 26
2.4 Videreudvikling 28
Evalueringsdata 28
Iterativ værdiskabelse 29
Udvidet understøttelse af SFU-samtalen 29
Komponentforbedringer 30
2.4 Idriftsættelse 30
Dataflow og integrationer 30
Sikkerhed 31
Ressourcebehov og skalerbarhed 31
Miljøopsætning 32
Tidslinje 32
3. Journalopsummering (UC2) 34
3.1 Formål og beskrivelse 34
Afgrænsninger 34
3.2 Teknisk løsning 34
Afprøvningsklient 36
Dataprocessering 39
Promptstruktur – opsummering af journaler 40
3.3 Evaluering 41
3.4 Videreudvikling 42
3.5 Idriftsættelse 42

Dataflow og integrationer 42
Sikkerhed 43
Kvalitetssikring 43
Tidslinje 44
4. Indtaling af besøgsbeskrivelser (UC3) 46
4.1 Formål og beskrivelse 46
4.2 Teknisk løsning 46
Afprøvningsklient 47
Opsummering og opdatering - Promptstruktur 53
4.3 Evaluering og afprøvning 54
Intern afprøvning 54
Ekstern afprøvning 54
Analyser 62
4.4 Videreudvikling 65
Hallucinationer 65
Forbedringer til struktur og kommune-tilpasning 66
Tekniske udviklingsmuligheder 66
Kvalitetssikring 67
Specialiserede muligheder for videreudvikling 67
4.5 Idriftsættelse 67
Miljøopsætning 67
Kvalitetssikring 68
Driftsmodeller 68
Sikkerhed 68
Organisatorisk forankring 69
Dataflow og integrationer 69
Ressourcebehov 69
Tidslinje 69
5. Bilag 70
A: Eksempel af visitations prompt 70
B: Eksempel af opdaterings prompt 71
C: Levenshtein distance for besøgsplaner 73

Figur-oversigt

Figur 1: Sygeplejefaglig udredning - Komponent oversigt 8
Figur 2a: Samtale optagelse 10
Figur 2b: Samtaleoversigt 10
Figur 2c: Evalueringsside for fagpersonalet 11
Figur 3: Pseudo-realtids-transskriberingsmodulet 13
Figur 4: Forudsigelsesstrategi med sliding window og majority voting 16
Figur 5: Flow for generering af sygeplejetilstandsnotater 17
Figur 6: 'LLM-as-a-judge' kvalitetsvurdering 23
Figur 7: Journalopsummering – Teknisk løsningsopbygning 35
Figur 8a: Oversigt over journaler 37
Figur 8b: Journalsiden for SPL-Use casen. 38
Figur 8c: Journaldata sektionen hvor de forskellige sygeplejetilstande ses 38
Figur 8d: Journalsiden for SSH/SSA-Use casen 39
Figur 9a: Feedback af LLM-generede opsummering for SPL-Use casen 41
Figur 9b: Feedback af manuelt generede opsummering for SPL-Use casen 42
Figur 10: Indtaling af besøgsbeskrivelse - Komponent oversigt 46
Figur 11a: Besøgsplanoversigten for en bruger 49
Figur 11b: Oprettelse af ny Besøgsplan 49
Figur 11c: Visning af den generede besøgsbeskrivelse. 50
Figur 11d: Indtal/indlæs en opdatering til besøgsbeskrivelsen 51
Figur 11e: Opdateringssiden 52
Figur 12: Fordeling af ratings for oprettede og opdaterede besøgsplaner 56
Figur 13: Boxplot for Levenshtein distance for oprettede og opdaterede besøgsplaner 57
Figur 14: Boxplots over udførselstider 62

Tabel-oversigt

Tabel 2: Oversigt over talegenkendelseskomponent versioner 12
Tabel 3: Oversigt over metoder og modeller 22
Tabel 4: Relevansklassificering 23
Tabel 5: Resultater fra UC1 afprøvninger 26
Tabel 6: Oversigt over fejltyper UC1 afprøvning 2 27
Tabel 7: Oversigt over antallet af oprettede og opdaterede besøgsplaner 55
Tabel 8: Gennemsnitlig rating for afprøvning for hver kommune. 55
Tabel 9: Typer af fejl i oprettede besøgsplaner med rettelser fortaget (76 i alt) 58
Tabel 10: Temaer i feedback tekster på oprettede besøgsplaner 59
Tabel 11: Temaer i feedback tekster på opdaterede besøgsplaner 59
Tabel 12: Beregnings og udførselstider på "Indtaling af besøgsbeskrivelser" 60
Tabel 13: Dage for afprøvning af Use case 3 60
Tabel 14: Andelen af besøgsplaner med hallucinationer (%) 65
Tabel 15: Levenshtein distance mellem oprettede / opdaterede og rettede besøgsbeskrivelser 73

1. Introduktion

Dette dokument er en større teknisk redegørelse for de løsninger udviklet og afprøvet i projektet. Dokumentet har til formål at besvare følgende spørgsmål:

- Hvad har vi bygget?
- Hvor godt virkede det?
- Hvordan kan man forbedre på løsningerne?
- Hvad skal man være opmærksom på i vejen mod drift?

Vi har i projektet arbejdet med 3 områder for værdiskabelse i sygeplejen, kaldet Use cases, og har til hver af dem udviklet en løsning baseret på talegenkendelse, tekstklassificering og/eller generative tekst modeller. Løsningerne har det til fælles at de forsøger at afhjælpe byrden forbundet med at generere eller orientere sig i journaldata. Disse løsninger har vi gennem afprøvninger vist frem til sygeplejefagligt personale fra et bredt udsnit af kommuner.

Projektet har arbejdet med forskellige muligheder for værdiskabelse i syge- og hjemmeplejen samlet i 3 Use cases (UC). Følgende er Use cases overskrift, efterfulgt af en kort beskrivelse:

- **UC1: Sygeplejefaglig udredning**

Støtte til oprettelse af sygeplejetilstandsnotater på baggrund af en sygeplejefaglig udredningssamtale (SFU-samtale) mellem borger og sygeplejerske. Støtten ydes gennem udkast til notater baseret på samtalens indhold.

- **UC2: Opsummering af borgerjournal**

Opsummering af borgerjournal til støtte ved borgerbesøg udført af hhv. Sygeplejersker (SPL) og sundheds- og omsorgsassistenter og -hjælpere (SSA/SSH).

- **UC3: Indtaling af besøgsbeskrivelse**

Støtte til oprettelse og opdatering af besøgsbeskrivelser til hjælp med hjemmebesøg udført af SSA/SSH. Besøgsbeskrivelserne oprettelse på baggrund af en indtaling.

For mere information om arbejdet med at definere, afgrænse og afprøve projektets Use cases se dokument "Talt: Uses cases og afprøvning fra et sundhedsfagligt perspektiv"

2. Sygeplejefaglig udredning (UC1)

2.1 Formål og beskrivelse

Når en borger modtager sygeplejeydelser, foretages en sygeplejefaglig udredning, hvor borgeren vurderes på de 12 sygeplejefaglige problemområder (SPO). Dette dokumenteres i borgerens elektroniske journal på kortet sygeplejetilstande fra Fælles Sprog 3.

Den del af den sygeplejefaglige udredning der undersøges i dette projekt, er sygeplejefaglig udredning ved nye borgere, der har behov for sygeplejeydelser.

Når en borger skal opstarte med sygeplejeydelser, skal der indledningsvist foretages en sygeplejefaglig udredning, hvor i der indgår en vurdering de 12 sygeplejefaglige problemområder.

En sygeplejefaglig udredning er en gennemgang af borgerens helbred og afdækning af de sygeplejefaglige problemområder. Udredningen har til formål at afdække behovet for sygeplejefaglig behandling. Udredningen gennemføres ved at tale med borgeren om de helbredsudfordringer, der ligger til grund for henvendelsen, det behov, der er for sygeplejefaglig behandling.

Efter samtalen skal sygeplejersken dokumentere i borgerens journal og beskrive borgerens helbredsproblem på de aktuelle sygeplejetilstande under Fælles Sprog 3. Det er denne dokumentation, der ligger til grund for de sygeplejefaglige indsatser der efterfølgende sættes i værk.

Der kan være latenstid for dokumentationen, da sygeplejersken kan have andre opgaver,fa der skal varetages, og det er derfor en mulig årsag til fejl i den sygeplejefaglige udredning. Ydermere er sygeplejefaglig udredning en lang proces med mange tidskrævende skridt grundet den store mængde af dokumentation der skal udarbejdes.

Den sygeplejefaglige udredning og efterfølgende dokumentation på sygeplejetilstande under Fælles Sprog 3, er fælles dokumentationspraksis for alle kommuner i landet.

Det er derfor en oplagt Use case at undersøge ifht. brug af AI; de mange manuelle skridt samt den forsinkede journalisering kan være årsag til fejlkilder,I der kan minimeres med AI, og en forbedring/effektivisering vil have en stor effekt grundet den store udbredelse og store indsats det er at skrive dokumentationen. I denne Use case forsøger vi derfor at afklare følgende:

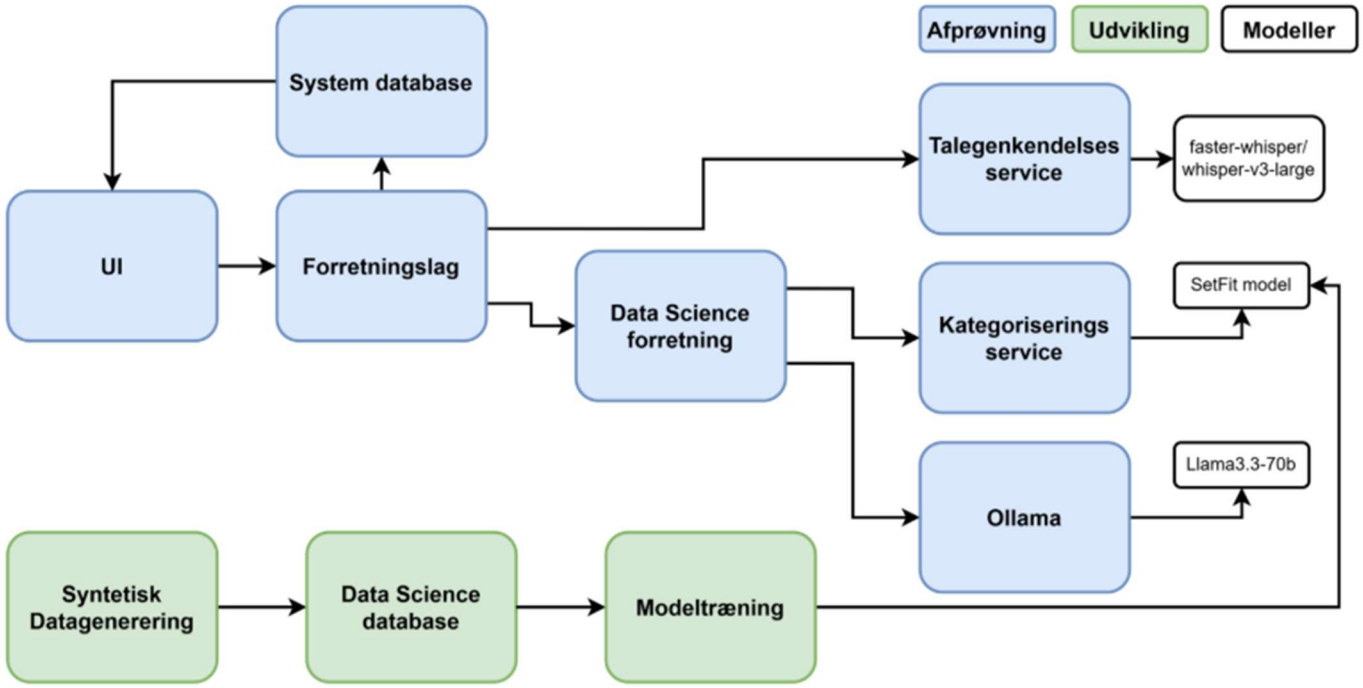
- *Hovedmål: Er det muligt for en AI-løsning at lave et brugbart udkast til dokumentationen for en sygeplejefaglig udredning, ved brug af en optagelse af en samtale mellem sygeplejersken og borgeren?*

2.2 Teknisk løsning

Use casen understøttes ved at SFU-samtalen optages og efterbehandles i vores afprøvningsklient, udkast til journalnotater genereres ved at samtalen transskriberes via en talegenkendelsesmodel (TGK), den resulterende tekst klassificeres på de 12 sygeplejefaglige problemområder, hvorefter relevante sygeplejetilstandsnotater genereres via en stor sprogmodel.

En række komponenter er udviklet til at opnå ovenstående løsning og for at sikre at projektets behov for udvikling, afprøvning og dataindsamling kan understøttes af den samlede tekniske løsning. Se figur 1 for et samlet overblik over komponentarkitekturen for løsningen.

Figur 1: Sygeplejefaglig udredning - Komponent oversigt



Komponenter markeret i blåt udgør løsninger. Grønne komponenter er brugt til udvikling af løsningen. Hvide komponenter er modeller, både anvendte og trænede.

Følgende er kort beskrivelse af de centrale komponenter:

- **UI:** Afprøvningsklientens brugergrænseflade, udstilling af løsningen til brugere i forbindelse med afprøvning samt interne projektfolk i forbindelse med test og udvikling.
- **Forretningslag:** Afprøvningsklientens forretningslag, til understøttelse af brugergrænsefladen og sammenkobling mellem system og database.
- **Data Science forretning:** Koblingspunkt mellem afprøvningsklienten og Data Science komponenterne samt forretningslogik.
- **Talegenkendelsesservice:** API til udstilling af Whisper-model.
- **Kategoriseringsservice:** API til udstilling af kategoriseringsmodel
- **Ollama:** API til udstilling af sprogmodeller
- **Syntetisk datagenerering:** Modul til generering af syntetiske SFU-samtaler brugt til udvikling af Data Science komponenterne. Læs mere om dette modul i supplerende dokument 'Syntetiske data'.

Du kan læse mere om modeller trænet og data indsamlet i forbindelse med udvikling og afprøvning af løsningen i supplere dokument 'Data og modeller'.

Afprøvningsklient

Afprøvningsklienten er udviklet som et evalueringsværktøj til fagpersonale, der skal teste systemets evne til at generere sygeplejetilstandsnotater baseret på en sygeplejefaglig udredningssamtale af en borger. I de følgende afsnit beskrives klientens brugergrænseflade, funktionalitet og de designovervejelser, der ligger til grund for løsningen.

Brugergrænseflade

Afprøvningsklienten er bygget til at understøtte afprøvning af Use casen, hvis primære formål, er at gøre det muligt for fagpersonale at teste kvaliteten på løsningens samlede output. Derfor understøtter brugergrænsefladen følgende funktionalitet:

- Oprettelse af samtaler med valg af testborgere
    - Realtidsindtaling af samtale
    - Upload af optaget samtale
- Evaluering af udkast til sygeplejetilstandsnotater
    - Redigering af udkast
    - Oprettelse af notat til missede sygeplejetilstande
    - Godkendelse af endelige notater

Klientbeskrivelse

Når brugeren opretter en ny samtale, skal der vælges, hvilken borger samtalen skal oprettes på. Man kan vælge at indlæse en eksisterende samtale eller lave en realtidssamtale, hvor man indtaler en dialog mellem bruger og borger jf. figur 2a. Hvis man laver en realtidssamtale, så vil systemet transskribere samtalen løbende, og dermed vil belastningen på systemet være fordelt over en længere tidsperiode. Realtidstransskriberingen foregår ved at brugergrænsefladen sender lydbidder hver 5 minut, som kan justeres efter behov. Når samtalen afsluttes, sendes et slutsignal. Systemet venter på, at alle lydbidder er transskriberet, hvorefter de samles til én samlet transskribering, der sendes til kategoriseringsmodellen. Brugeren vil blive navigeret ud til samtalelisten, og herfra kan brugeren lave en ny samtale eller kigge på tidligere samtaler.

Færdigprocessering af samtalen foregår asynkront med brugergrænsefladen. Brugeren vil se, at samtalens status er gul, hvilket indikerer, at den enten er i kø eller under processering. Når samtalen er færdigprocesseret, markeres den med grønt. Det er brugerens ansvar at tjekke, hvornår processering er afsluttet ved at trykke på opdater-knappen. Der er bevidst ikke implementeret automatisk genopfriskning for at spare ressourcer. Se figur 2b.

I samtaleoversigten kan man vælge samtaler, der er færdigprocesseret og klar til evaluering. Når man trykker på en samtale, vil man blive navigeret ind på en side, hvor man bliver præsenteret for to sektioner – relevante områder og ikke-relevante områder, der indkapsler alle mulige sygeplejetilstande jf. figur 2c.

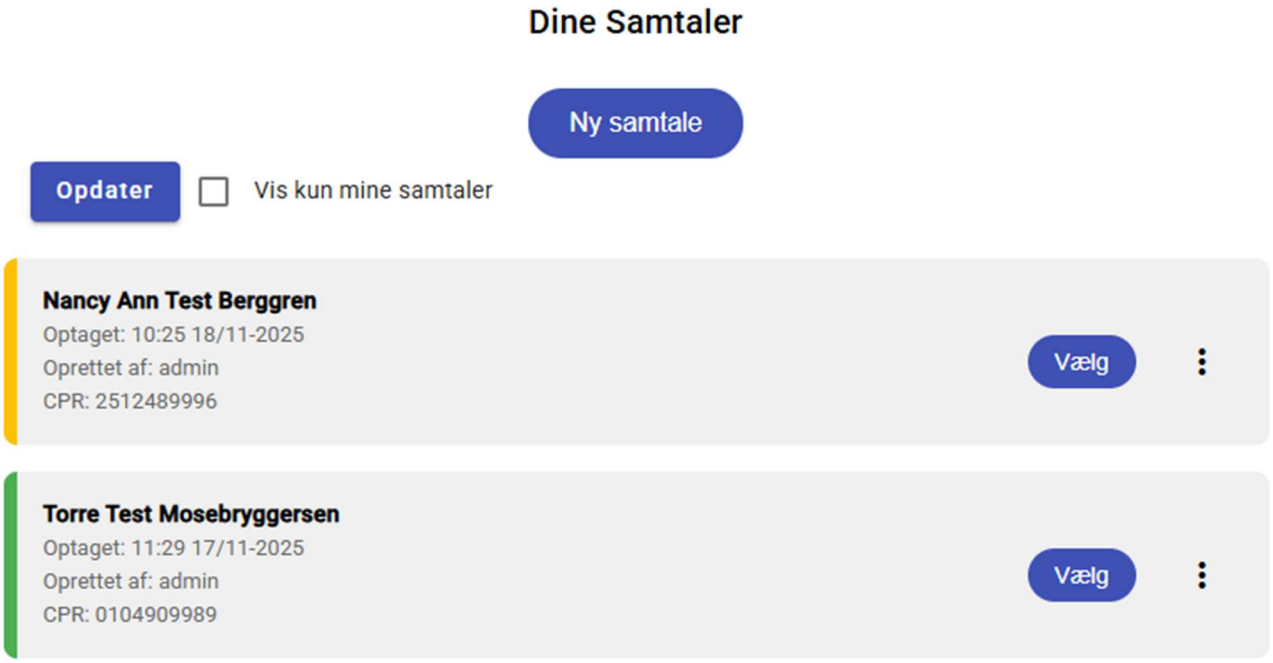
Systemet har delt samtalen op i relevante og ikke-relevante sygeplejetilstande ved at generere et udkast til sygeplejetilstandsnotater for de tilstande, den vurderer relevante for den givne borger. Hvis tilstanden ikke er relevant, skal brugeren markere den som ikke-relevant, så tilstanden flyttes til ikke-relevante områder. Omvendt, hvis systemet har misset en relevant tilstand, kan brugeren tilføje tilstanden som relevant og lave et sygeplejetilstandsnotat. For hver tilstand kan man se kontekst fra samtalen og dermed tage stilling til, om tilstanden er relevant eller ikke-relevant.

Hvis tilstanden er relevant, forventes det, at brugeren tager stilling til udkastet til det sygeplejetilstandsnotat. Brugeren skal redigere notatet, hvis det er misvisende eller mangelfuldt, og slutteligt godkende det. Der er også mulighed for, at brugeren kan tilføje en tekst om risikoen for tilstanden. Når brugeren har taget stilling til alle relevante tilstande, er evalueringen slut.

Figur 2a: Samtale optagelse

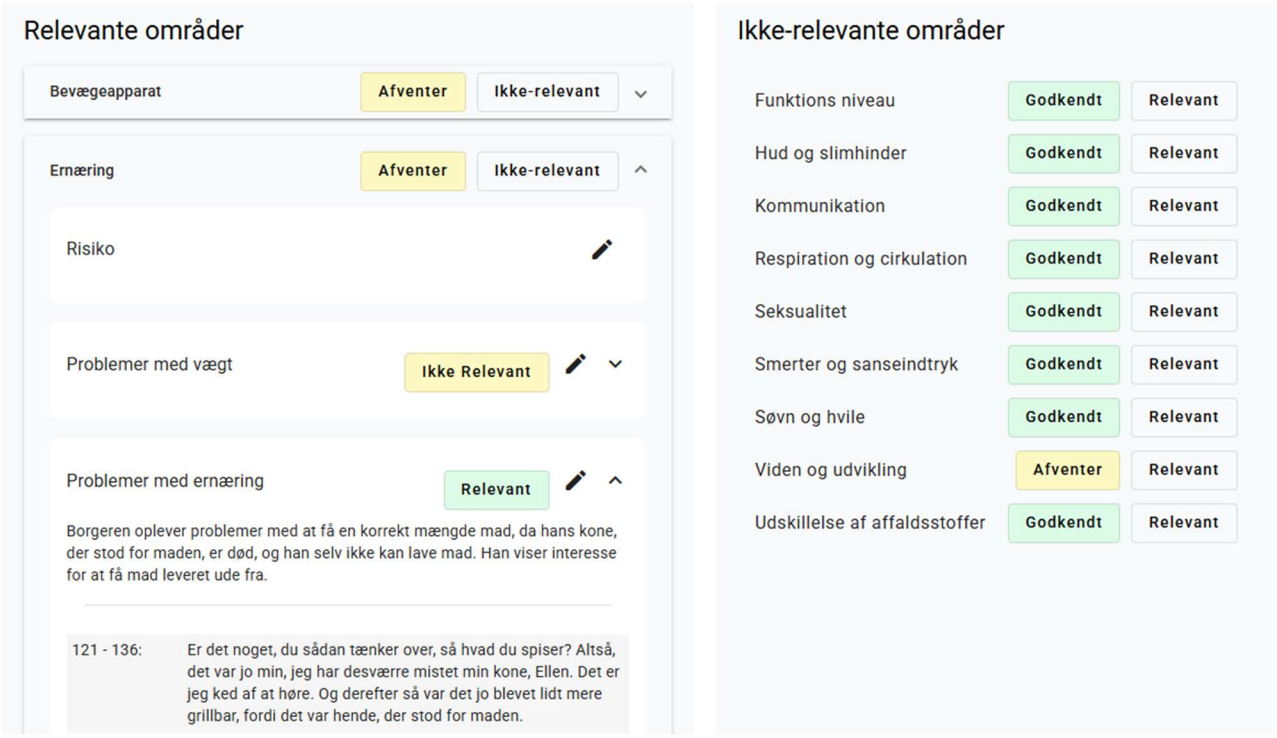
Brugeren vælger en borger, og indlæser dernæst en lydfil eller starter en realtidssamtale. Når optagelse er påbegyndt vises varigheden til brugeren.

Figur 3b: Samtaleoversigt



Gul markerer, at systemet arbejder. Grøn betyder, at samtalen er klar til evaluering.

Figur 4c: Evalueringsside for fagpersonalet



Under ernæringsområdet, ses tilstanden "problemer med ernæring" og dens journalnotat, som er en opsummering af samtaleemner omhandlende ernæring.

Designovervejelser

Da systemet skal kunne håndtere flere brugere på samme tid, er det essentielt, at brugeren ikke bliver låst, mens systemet arbejder, da det kan tage lang tid at processere en samtale. Derfor er klienten asynkron, og brugeren er fri til at lave andet, mens transskriberingen, kategoriseringen og opsummeringen er i gang. Det er en fordel i afprøvningsøjemed, da brugeren kan oprette andre samtaler eller evaluere samtaler, mens tidligere samtaler processeres.

I designprocessen har der desuden været fokus på, at brugeren skulle finde brugergrænsefladen overskuelig. En samtale kan være lang, og der kan være mange referencer til samtalelinjer. Derfor er brugergrænsefladen lavet med flere paneler, der kan foldes ud og ind, således dataet er gemt væk, se Figur 2c. De tilstande, som brugeren skal tage stilling til, bliver vist i venstre side, og de ikke-relevante kan ses i højre side. Brugeren kan nemt markere et område som ikke-relevant ved at trykke på ikke-relevant-knappen og omvendt. Alle ændringer – nye statusser, redigeringer eller godkendelser – gemmes øjeblikkeligt på serveren. Dette er en bevidst designbeslutning, der gør det muligt at indsamle feedback på systemets præcision uden at implementere en separat feedbackmekanisme. Systemet gemmer både det originale og det rettede sygeplejetilstandsnotat, hvilket muliggør analyse af, i hvor høj grad notaterne kræver manuel gennemgang.

Udover disse designvalg findes der også elementer i brugergrænsefladen, som afspejler løsningens udvikling gennem projektforløbet. Der findes elementer i brugergrænsefladen, som stammer fra funktionalitet, der blev planlagt, men ikke implementeret. I begyndelsen af designfasen blev det fastlagt, at systemet skulle fokusere på at automatisere dokumentationsarbejdet omkring sundhedsfaglige udredninger. Planen var, at sygeplejetilstandsnotater skulle genereres automatisk ud fra samtalen med borgeren og efter faglig gennemgang sendes direkte til et EOJ-system som Columna

Cura, hvor de kunne integreres i borgerens journal. Dog viste det sig i løbet af projektforløbet, at EOJ-integrationen var en større opgave end forventet, og at den ikke skabte tilstrækkelig værdi i forhold til at teste systemets kerneteknologi. Derfor blev konceptet skrottet, men der er stadig spor af det i brugergrænsefladen. F.eks. findes der en "journaliser"-knap, som brugeren skal trykke på, når alle tilstande er godkendt – en funktion der oprindeligt var tiltænkt at sende data til EOJ-systemet, men som ikke er implementeret.

På trods af ændringer i løsningens funktionalitet har én designfilosofi været gennemgående i hele processen. Til sidst skal der nævnes, at human-in-the-loop har været en kernetanke, og det er derfor, konceptet med godkendt/afventer findes på alle tilstande. Det er vigtigt at understrege, at systemet laver udkast til sygeplejetilstandsnotater, og det er brugerens ansvar at godkende alle udkast, før det endelige notat bliver genereret.

Talegenkendelse

Flere af projektets tekniske løsninger er baseret på tale som input data, vi har derfor, i løbet af hele projektet, arbejdet med en komponent til talegenkendelse (TGK), hvis formål er at oversætte tale-til-tekst. Denne komponent er primært udviklet i forbindelse med løsningen til sygeplejefaglig udredning og derefter benyttet til indtaling af besøgsbeskrivelser. Udvikling på komponenten er foregået løbende, se nedenstående tabel 2 for en oversigt over de primære versioner.

Tabel 1: Oversigt over talegenkendelseskomponent versioner

Version	Beskrivelse	Whisper model	Præprocessering	Processeringstid (30 min. tale)
1	Selvhostet API baseret på 'automatic-speech-recognition' pipeline fra Transformers	Whisper-v3-large	Filkonvertering	Ca. 7 minutter
2	Eksternt hostet API hos Cloudflare	Whisper-v3-turbo	Filkonvertering, chunking (30 s)	Ca. 5 minutter
3	Selvhostet API baseret på faster-whisper fra SYSTRAN	Whisper-v3-large	Filkonvertering, chunking (300 s), pseudo-realtid	Ca. 2 minutter

Selve TGK-komponenten står for oversættelsen af lyd til tekst. Det har fra projektets start været en ambition at kunne hoste denne del på eget hardware. Version 1 og 3 opfylder denne ambition. Undtagelsesvis blev version 2 implementeret der benyttede et ekstern API til første afprøvning af systemet for at opnå den nødvendige skalering.

TGK-komponenten kan dog sjældent stå alene, når den implementeres i en løsning. Derfor er en række præprocesserings og understøttelses moduler implementeret. Nedenstående er en liste af moduler med tilhørende beskrivelse:

1. Filkonvertering

Både ved optagelse af lyd eller upload af lydfiler i en løsning, skal lyden konverteres til det optimale format. Vi benytter .wav som format for vores lyd.

2. Chunking

En optagelse af lyd eller uploadet lydfil inddeles i bidder af en given længde. Dette er påkrævet af forskellige årsager, det eksterne API i version 2 havde en begrænset filstørrelse og realtidstransskriberingen kræver at lyden deles op, mens den optages. Ideelt set har denne inddeling af lyden minimal indflydelse på kvaliteten af transskriptionen, dette opnås bedst ved at inddele lyden ved naturlige pauser i samtalen, hvorfor inddelingen med fordel kan udføres ved større perioder med stilhed. Dette kaldes 'silenced based chunking'.

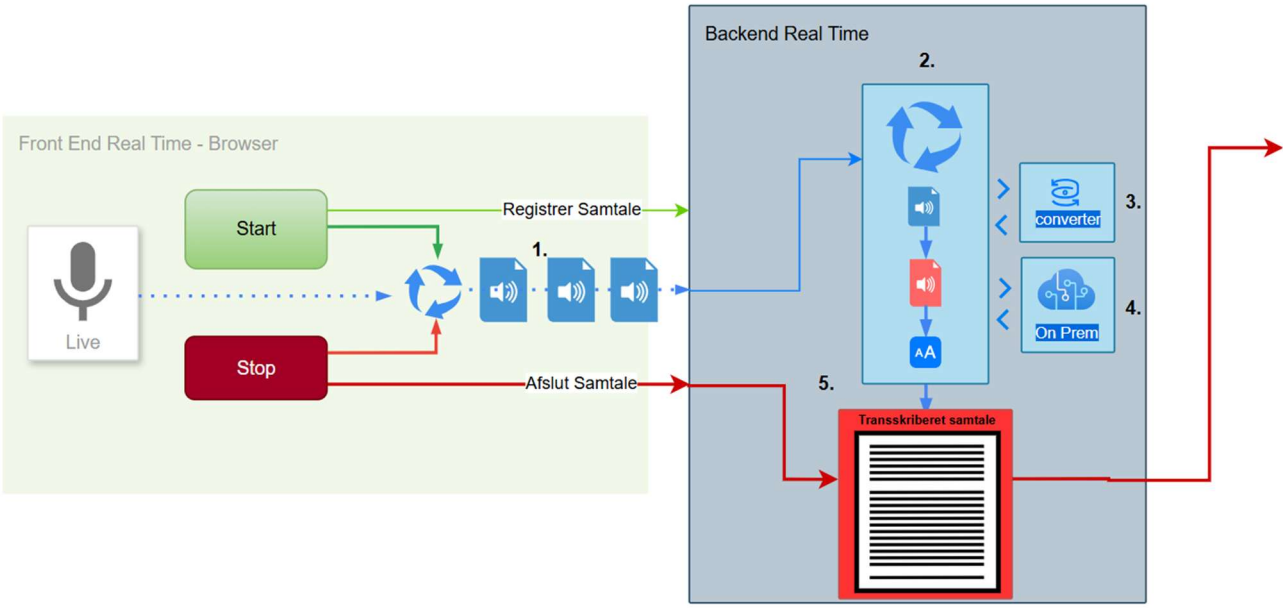
3. Realtidstransskribering

Transskriptionsprocessen igangsættes mens lyden optages, så den endelig transskription er klar umiddelbart efter optagelsen, er færdig.

Vi har implementeret dette ved at alle samtidige optagelser i løsningen løbende sender lydbidder til processering. Da disse lydbidder ikke tager højde for stilhed, er lydbids længden sat til 300 sekunder for at begrænse indflydelsen på transskriptionskvaliteten. Herfra konverteres lydbidder til dette rette format og den konverterede lydfil transskriberes. Alle disse skridt på tværs af alle igangværende optagelser håndteres af co-routines.

Når en given optagelse er færdig og alle lydbidderne er modtaget og transskriberet, sender finalize-co-routinen den komplette transskription videre i vores system.

Figur 5: Pseudo-realtids-transskriberingsmodulet



Arkitekturtegning for realtidstransskriberingsmodulet.

Se figur 3 for et diagram der illustrerer flowet. "Converter"-rubrikken symboliserer konverteringen til .wav, og "On Prem"-rubrikken symboliserer TGK-komponenten.

1. Indkommende lydbidder
2. Kø-systemerne
3. Konvertering til WAV-lyd
4. Transskription via TGK-komponent
5. Stop-signal – sender sidste lydbid med et tag, der aktiverer finalize-coroutinen.

Den endelige løsning for TGK-komponenten og dens implementering i løsningen til UC1 er altså transskription med Whisper-v3-large implementeret via faster-whisper, udstillet i et FastAPI modul, hostet på eget hardware. Selve processen for optagelse og transskription af flere samtidige samtaler bliver orkestreret af realtidstransskriberingsmodulet implementeret via co-routines. Modulet gør brug af et filkonverteringsmodul der sikrer ensartet behandling af lyden i løsningen.

Størstedelen af arbejdet for at sikre transskribering af tale-til-tekst i løsningen har ikke været selve komponenten, hvor Whisper-v3-large, tidligt i projektet, blev vurderet til at være af tilstrækkelig kvalitet til at understøtte Use casen. I stedet har fokus ligget på den omkringliggende orkestrering og skalering af løsningen til flere samtidige optagelser.

For at skabe de nødvendige rammer for at kunne udføre afprøvninger skulle systemet gerne have et svar indenfor 15 minutter (efter endt optagelse) med op til 10 samtidige brugere. Dette understøttede version 1 af komponenten på ingen måde, da samtidige optagelser ville blive behandlet sekventielt, hvilket i værste fald betød at 10 samtidige brugere kunne betyde at enkelte brugere ville opleve en svartid, langt, over den aftale grænse.

Derfor blev version 2 midlertidigt implementeret i forbindelse med den første afprøvning af use casen. Her benyttede vi en eksternt hostet forbrugsbaseret endpoint hos Cloudflare til transskription og kunne derfor transskribere flere samtidige filer, hvorfor systemets kapacitet for flere samtidige brugere ikke længere var begrænset af transskriberingen. Endpointet havde en størrelsesbegrænsning på mængden af lyd der kunne processere per kald, hvorfor vores 30 minutters lyd-filer måtte chunkes. En chunkstørrelse på 30 sekunder blev valgt og transskriberingen led derfor et større kvalitetstab sammenlignet med den lokale version.

Version 3 blev planlagt som en version af komponenten der kunne understøtte det nødvendige load ifm. Afprøvning på eget hardware. Vi blev i mellemtiden mere bevidste om i hvor stor grad chunking med en længde på 30 sekunder påvirkede kvaliteten på transskriberingen. I en ideel verden var silenced-based-chunking blevet implementeret, men dette blev ikke muligt grundet tidspres. Derfor blev chunk længden sat op til 300 sekunder, for at minimere tabet af kvalitet.

Kategorisering

I understøttelsen af UC1 valgte vi at støtte opsummeringen af den lange udredningssamtale til de 23 sygeplejetilstandsnotater ved at identificere relevante sygeplejefaglige problemområder i den enkelte samtale og isolere den del af samtalen der er relevant for hvert problemområde. Denne inddeling opnås med en tekst klassifikationsmodel, trænet til det specifikke formål at klassificere mindre bidder af udredningssamtaler.

Da vi forventede at meget lidt data ville være tilgængelig til at træne denne model, valgte vi at bruge modelarkitekturen fra SetFit modulet[1]. Denne modelarkitektur udnytter en valgfri præ-trænet tekst embedding model og optimerer den til det valgte scenarie via 'supervised constrastive learning'. En proces hvor trænings eksempler dannes via par af tekststykker. Tekststykker indenfor samme kategori danner positive par, hvorimod tekststykker fra forskellige kategorier danner negative par. I løbet af træningen flyttes positive par tættere i embedding rummet og negative par flyttes længere fra hinanden. Denne modelarkitektur og tilhørende træningsmetode har potentiale for at opnå høj performance på relativt få eksempler.

Med opgaven defineret og modelarkitekturen valgt er følgende spørgsmål relevante at besvare:

[1] GitHub - huggingface/setfit: Efficient few-shot learning with Sentence Transformers

- Hvad er vores input?
- Hvilke data har vi til rådighed?
- Kan vi supplere modellen til at opnå bedre resultater?

Klassifikationsmodellen falder ind mellem talegenkendelsesmodellen og selve journalnotats genereringen. Det betyder at inputtet til modellen i sin rå form vil være transskriberingen af SFU-samtalen. Denne transskribering er inddelt i tekstbidder defineret af TGK-modellen, de varierer i størrelsen, men er oftest ret små. Grundet deres lille størrelse indeholder en given tekstbid ofte ikke nogen semantisk betydning for samtalens aktuelle emne. Derfor har vi behov for at sammenlægge flere tekstbidder i tekststykker (chunks) og derved højne sandsynligheden for at relevant semantisk betydning er til stede i et givent tekststykke. Denne sammenlægning kan udføres med et vælg af forskellige heuristikker.

Den heuristik, vi har valgt at bruge benytter to processer: sliding window og majority voting.

Sliding window

Hver linje sammenlægges med omkringliggende linjer i et forudsigelsesvindue, dette vindue centreres omkring en given linje og udfører en forudsigelse på det samlede tekststykke i vinduet. Dernæst rykker vinduet en linje ned og processen gentages. Vi definerer størrelsen ud fra antallet af linjer før og efter den aktuelle linje. Dette refereres til som 'padding' og padding størrelsen (p) definerer størrelsen på vinduet (n) som 2*p+1

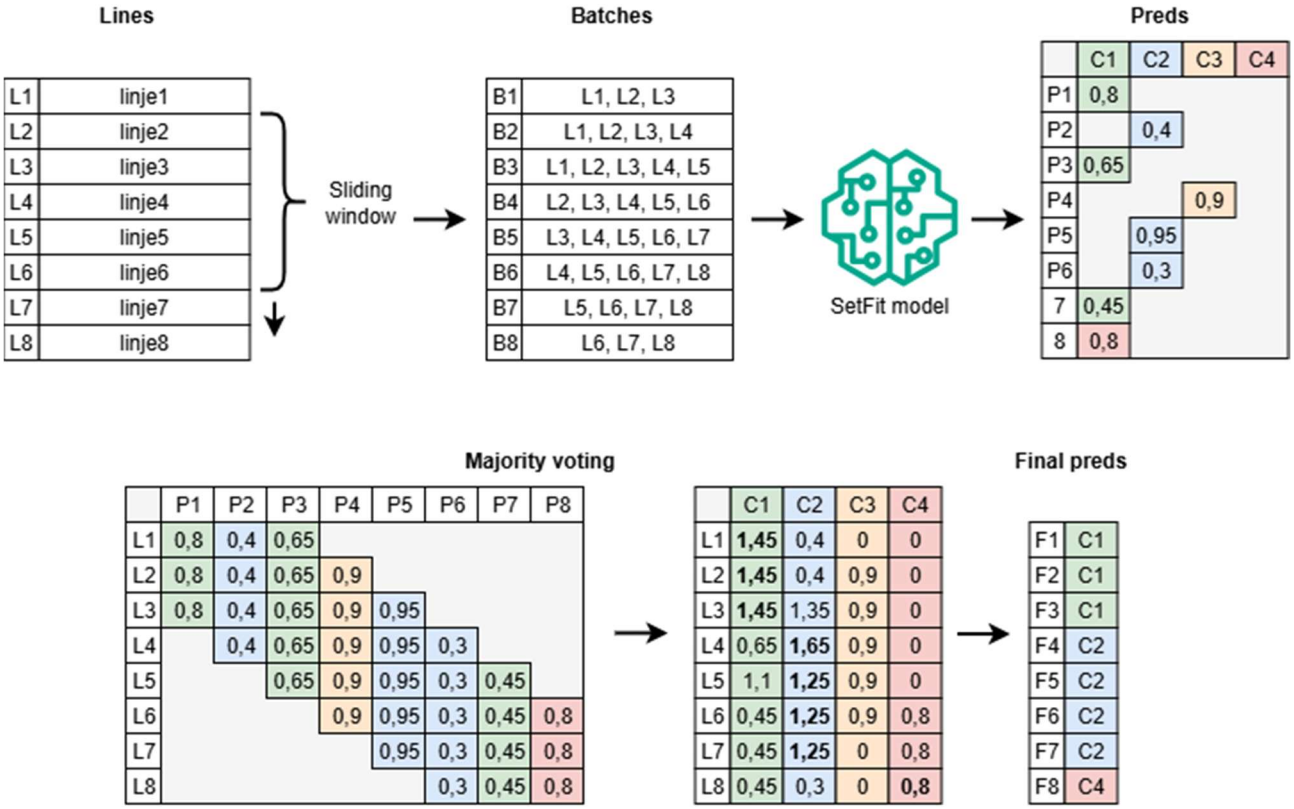
Majority voting

En liste af forudsigelser for den enkelte linje samles, både den forudsigelse hvor linjen er i fokus, samt de forudsigelser hvor linje indgår i padding til en anden linje. Denne liste har samme længde som vinduesstørrelsen (n). Hver af disse forudsigelser 'stemmer' med modellens confidence score (variant: forudsigelsen udgår én hel stemme) på en given kategori og den kategori med flest stemmer vinder.

Forudsigelsen for hver linje er defineret ud fra forudsigelsen for hver sliding window eksempel, hvor den givne linje enten er i fokus eller optræder i paddingen. Med en padding størrelse (p) på 2, bliver vinduets størrelse (n) lig med 5 (2*p+1), hvilket betyder at 5 forudsigelser forekommer for den enkelte linje. For de 5 forudsigelser lægges modellens confidence score sammen per kategori og den kategori med den højeste score udgår den endelige forudsigelse.

Se figur 4 for en illustration af forudsigelse med sliding window og majority voting.

Figur 6: Forudsigelsesstrategi med sliding window og majority voting



Visualisering af strategien til forudsigelse at sygeplejefaglige problemområder (illustreret med 4 kategorier) på linjer i sygeplejefaglige udredningssamtaler. Her er padding størrelsen (p) = 2, hvorfor vinduesstørrelsen (n) = 5.

Opsummering – Promptstruktur ("Hvordan har vi opsat vores prompting")

Opsummeringskomponenten har til formål at generere strukturerede sygeplejetilstandsnotater fra kategoriserede sundhedsfaglige udredningssamtaler. Komponenten modtager transskriberede SFU-samtaler, der allerede er kategoriseret til et specifikt sygeplejefagligt område (SPO) fra kategoriseringskomponenten. Herfra leverer opsummeringskomponent et JSON-objekt med sygeplejetilstandsnotater for alle relevante sygeplejetilstande under det givne SPO.

Promptarkitektur: Todelt struktur

For at kunne håndtere alle 12 SPO'er dynamisk, hvor hvert SPO indeholder mellem 1-4 sygeplejetilstande, er opsummeringskomponenten baseret på en todelt promptarkitektur.

Den første del, prompt intro, er en generel instruktion der forbliver identisk uanset hvilket SPO, der behandles. Den definerer AI-assistentens rolle som FSIII-specialist og beskriver den fundamentale proces:

1. Læsning og analyse af den kategoriserede sundhedsfaglige udredning
2. Identifikation af relevante sygeplejetilstande
3. Formulering af præcise notater for hver relevant tilstand
4. Returnering af output som JSON.

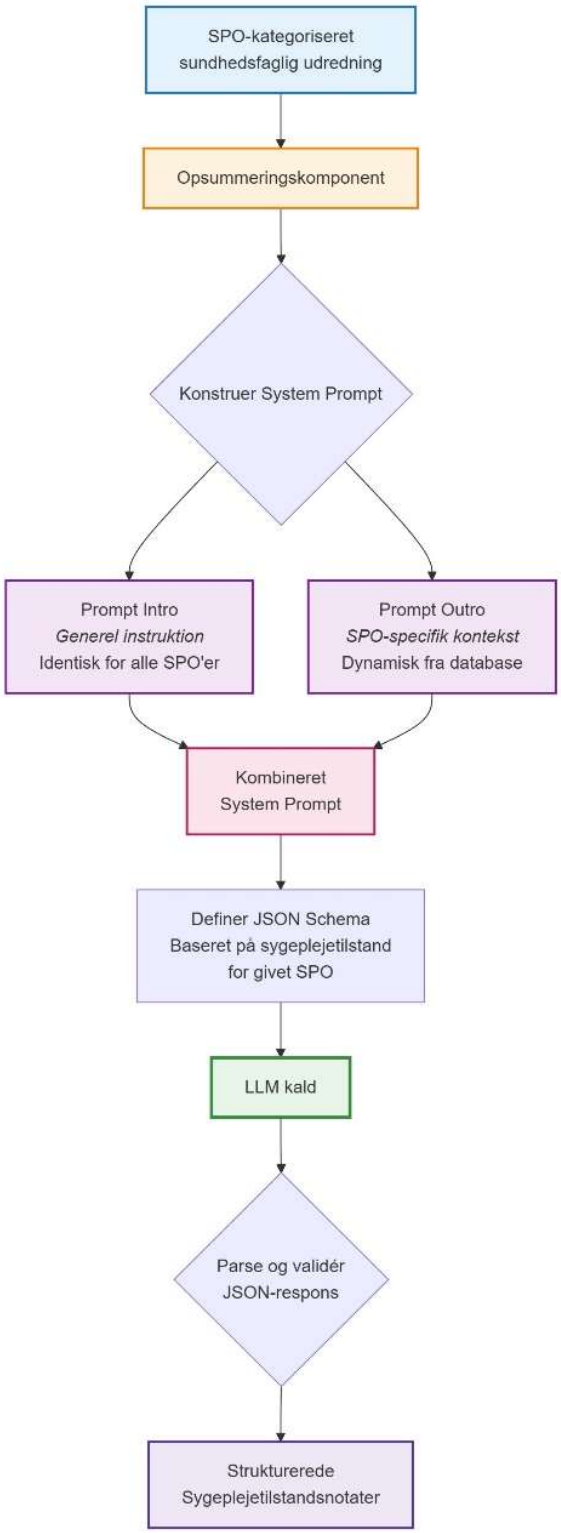
Prompt intro specificerer også, at der kun skal inkluderes relevante tilstande, samt at sproget skal være klart dansk med korrekt FSIII-terminologi og fokus på faktuel information.

Den anden del, prompt outtro, er den SPO-specifikke kontekstualisering, der genereres automatisk baseret på det aktuelle SPO. For hvert SPO hentes SPO-navn, en liste over alle sygeplejetilstande under området, beskrivelser af hver sygeplejetilstand, samt konkrete eksempler, der viser, hvordan en sundhedsfaglig udredning omskrives til et sygeplejetilstandsnotat. Denne dynamiske tilgang sikrer, at systemet skalerer til alle 12 SPO'er, og at nye områder eller tilstande nemt kan tilføjes.

Workflow for generering af sygeplejetilstandsnotater

Generering af sygeplejetilstandsnotater følger en struktureret proces i seks trin. Processen kombinerer de to dele af promptarkitekturen med dynamisk schema-generering og LLM-baseret tekstgenerering. Se nedenstående figur 5 for en illustration af hele processen.

Figur 7: Flow for generering af sygeplejetilstandsnotater



For at sikre konsistent og validerbart output anvendes dynamisk JSON schema-generering. Schema defineres ved runtime baseret på sygeplejetilstandene for det givne SPO, hvor hver sygeplejetilstand bliver en property i JSON-objektet. Systemet anvender OpenAI's structured output feature, som

garanterer at LLM-outputtet matcher det definerede schema og eliminerer behov for post-processing af malformed JSON.

Iterativ udvikling og forbedringer

Opsummeringskomponenten er blevet udviklet iterativt mellem to eksterne afprøvninger med fokus på at forbedre outputkvalitet og præcision gennem systematiske prompt-optimeringer.

1. Eksterne afprøvninger

Prompten ved 1. eksterne afprøvning anvendte den todelte promptstruktur beskrevet i afsnit 2. Selvom denne struktur etablerede det grundlæggende framework, viste afprøvningen, at LLM'en havde udfordringer med at producere output i den ønskede kvalitet og præcision. Output manglede ofte den specifikke information fra den sygeplejefaglige udredning eller inkluderede irrelevant information i de endelige sygplejetilstandsnotater.

Efter 1. afprøvning blev seks centrale problemområder identificeret, der påvirkede kvaliteten af de genererede sygeplejetilstandsnotater:

- Transskriberingsfejl overført til output
- Output på forkerte sprog
- Upræcise formuleringer
- Manglende information i opsummeringer
- Irrelevant information inkluderet
- Udledning af ikkeeksisterende information

Promptoptimeringsanalyse efter 1. afprøvning

Baseret på de identificerede problemområder blev der gennemført en systematisk afprøvning af forskellige prompt-variationer med henblik på at udvikle effektive mitigerings strategier. Hver variation fokuserede på at adressere specifikke problemer gennem ændringer i promptformuleringer og tilføjelse af eksplicitte instruktioner. Følgende strategier blev testet:

Sprogfokus: For at sikre dansk output selv når input indeholder fremmedsprog, blev der tilføjet eksplicit instruktion "Det skal altid være på dansk uanset sproget i samtalen".

Håndtering af stavefejl: For at forhindre at transskriberingsfejl kopieres til output, blev der inkluderet lister over typiske fejl og deres korrekte termer. Dette viste sig at kræve betydelig indsats, da fejlene skulle specificeres to gange i prompten for at have effekt.

Formulering: For at forbedre sproglig kvalitet blev der tilføjet instruktioner om sprogligt højt niveau. Dette havde minimal effekt, da modellen har begrænsede evner på dette område. Et forsøg med efterfølgende sproglig korrektion blev fravalgt, da det reducerede præcisionen af indholdet.

Opfattelse af information: For at sikre at relevant information ikke mangler i sygeplejetilstandsnotaterne, blev prompten udvidet med en detaljeret liste over informationstyper der skal inkluderes: observationer, symptomer, udfordringer, medicin, behandlinger, ønsker, bekymringer, reaktioner, historik og målinger.

Filtrering af irrelevant information (mest effektiv): For at reducere antallet af irrelevante notater, blev der tilføjet klarere definition af, hvornår information er relevant samt instruktion om at returnere "ikke relevant", når information mangler. Dette gav den mest signifikante forbedring med 12% reduktion i false positives.

Håndtering af udledning: For at forhindre at modellen inkluderer information, der ikke kan udledes fra teksten, blev der tilføjet eksplicit instruktion om objektive beskrivelser med fokus på kun aktuelle problemer og faktuel information.

De iterative forbedringer viste, at mens nogle problemområder kunne adresseres effektivt gennem prompt-engineering (særligt filtrering af irrelevant information), havde andre områder mere begrænsede forbedringer (sproglig kvalitet) eller krævede uforholdsmæssig stor indsats (stavefejlshåndtering). Den kombinerede tilgang med alle effektive strategier gav den bedste samlede præstation.

2. Eksterne afprøvning

Den opdaterede promptstruktur ved 2. eksterne afprøvning integrerer alle de mitigeringsstrategier, der viste sig effektive ved ovenstående analyse. Prompt intro er blevet væsentligt udvidet med detaljerede instruktioner om informationstyper, eksplicit vejledning om dansk sprogbrug, klarere definitioner af objektivitet og faktuel information, samt konkrete eksempler på hvad beskrivelserne kan omfatte.

Prompt outtro følger samme struktur som ved 1. eksterne afprøvning med dynamisk genereret SPO-specifik kontekst, men er nu suppleret med de skærpede instruktioner fra den opdaterede intro, der sikrer mere præcist og relevant output. Særligt fokuseres der på at filtrere information korrekt, så kun relevant information for den specifikke helbredstilstand inkluderes.

Analyser

For at afsøge forbedringsmuligheder for løsningen gennemførte vi en systematisk evaluering af fem alternative genereringsmetoder. Det primære forskningsspørgsmål var: Kan vi forbedre kvaliteten af automatisk genererede sygeplejetilstandsnotater ved at udforske metoder uden begrænsninger på hardware eller eksekveringstid?

Evalueringen byggede på data fra 2. eksterne afprøvning, som omfattede 54 sygeplejefaglige udredninger fra 11 kommuner med i alt 1.242 sygeplejetilstande fordelt på 23 kategorier inden for 12 sundhedsproblemområder efter FSIII-standarden. De genererede sygeplejetilstandsnotater blev sammenlignet både med information fra de oprindelige sundhedsfaglige udredninger og med de manuelt reviderede notater fra afprøvningen.

Metoderne blev evalueret på tre dimensioner. Først undersøgte vi relevansvurdering ved at sammenligne, hvor godt hver metode identificerer relevante sygeplejetilstande sammenlignet med markeringerne fra afprøvningen. Dernæst anvendte vi LLM-as-a-Judge (DeepEval[2] via GPT-4o fra OpenAI[3]) til at vurdere notaternes kvalitet på seks dimensioner, via LLM-metrikker[4]: *coherence*, *correctness*, *faithfulness*, *grammatical correctness*, *professionalism* og *concision*. Endelig foretog vi en uddybende semantisk analyse af de mest lovende metoder for at undersøge, hvor godt de håndterer medicinsk terminologi gennem entitetsudtrækning.

De fem metoder

Afprøvningssetuppet anvender en to-trins pipeline, hvor modellen (sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2[5]) med logistisk regression først kategoriserer samtalelinjer til 12 sundhedsproblemområder via sliding window med majority voting, hvorefter Llama-3.3-70B-Instruct[6] genererer de strukturerede sygeplejetilstandsnotater.

De fem alternative metoder adresserer forskellige hypoteser om, hvordan denne pipeline kan forbedres. Se tabel 3 for en oversigt over metoder og nedenstående for en uddybende forklaring:

Metode 1 (Delopsummeringer) tilføjer et mellemtrin efter kategoriseringen: o3 sprogmodellen fra Open AI[7] laver først delopsummeringer per område som et støjfilter, før de endelige notater genereres. Hypotesen er, at opgaven bliver lettere, når indholdet først er filtreret og kondenseret.

Metode 2 (One-Shot) tager den modsatte tilgang ved helt at springe kategoriseringstrinnet over. Her mapper o3 direkte fra samtale til notater - enten med 23 separate kald per sygeplejetilstand eller ét samlet kald. Hypotesen er, at modellen dermed kan se sammenhænge mellem sygeplejetilstande, som risikerer at gå tabt i den opdelte pipeline.

Metode 3 (Alternative kategoriseringsmodeller) anvender en større embedding-model (intfloat/multilingual-e5-large-instruct[8]) for at reducere kategoriseringsfejl, der forplanter sig videre i pipelinen.

[2] OpenAI | DeepEval by Confident AI - The LLM Evaluation Framework
[3] GPT-4o Model | OpenAI API
[4] Introduction to LLM Metrics | DeepEval by Confident AI - The LLM Evaluation Framework
[5] sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2 · Hugging Face
[6] meta-llama/Llama-3.3-70B-Instruct · Hugging Face
[7] o3 Model | OpenAI API
[8] intfloat/multilingual-e5-large-instruct · Hugging Face

Metode 4 (LLM Sliding Window) går et skridt videre ved at erstatte embeddings helt med o3, som kan håndtere sproglige nuancer og kontekst bedre.

Metode 5 (Større LLM til opsummering): Forbedret opsummering via state-of-the-art modeller (5.1 med GPT-4.1[9] og 5.2 med o3) gennem øget modelkapacitet og bedre medicinsk domæneviden kan forbedre notaternes kvalitet alene.

Tabel 2: Oversigt over metoder og modeller

Metode	Kategoriseringsmodel	Opsummeringsmodel	Bemærkninger
Afprøvningssetup (baseline)	sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2 + logistisk regression	Llama-3.3-70B-Instruct	Sliding window + majority voting
Metode 1: Delopsummeringer	sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2 + logistisk regression	o3 (interim) → o3 (final)	Tre-trins pipeline
Metode 2.1: One-Shot (per tilstand)	Ingen (direkte mapping)	o3	23 kald per samtale, 2 eksempler
Metode 2.2: One-Shot (per samtale)	Ingen (direkte mapping)	o3	1 kald per samtale, 1 eksempel
Metode 3.1: Alt. Kategorisering - alle kategorier	intfloat/multilingual-e5-large-instruct + logistisk regression trænet på alle sygeplejetilstande	Llama-3.3-70B-Instruct	Alle sygeplejetilstande
Metode 3.2: Alt. kategorisering - uden andet	intfloat/multilingual-e5-large-instruct + logistisk regression trænet på sygeplejetilstande uden andet	Llama-3.3-70B-Instruct	Uden "andet"-kategori
Metode 4: LLM Sliding Window	o3 (sliding window + majority voting)	Llama-3.3-70B-Instruct	LLM til kategorisering
Metode 5.1: Større LLM	sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2 + logistisk regression	GPT-4.1	SOTA-model
Metode 5.2: Større LLM	sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2 + logistisk regression	o3	SOTA-model

Resultater

Se tabel 4 for en oversigt over metodernes evne til at udføre relevansvurderingen, der måler metodernes evne til at skelne mellem relevante og irrelevante sygeplejetilstande. Her performer afprøvningssetuppet bedst (F1=0.79), tæt fulgt af metode 1 (Delopsummeringer) (0.74), metode 5.2

[9] GPT-4.1 Model | OpenAI API

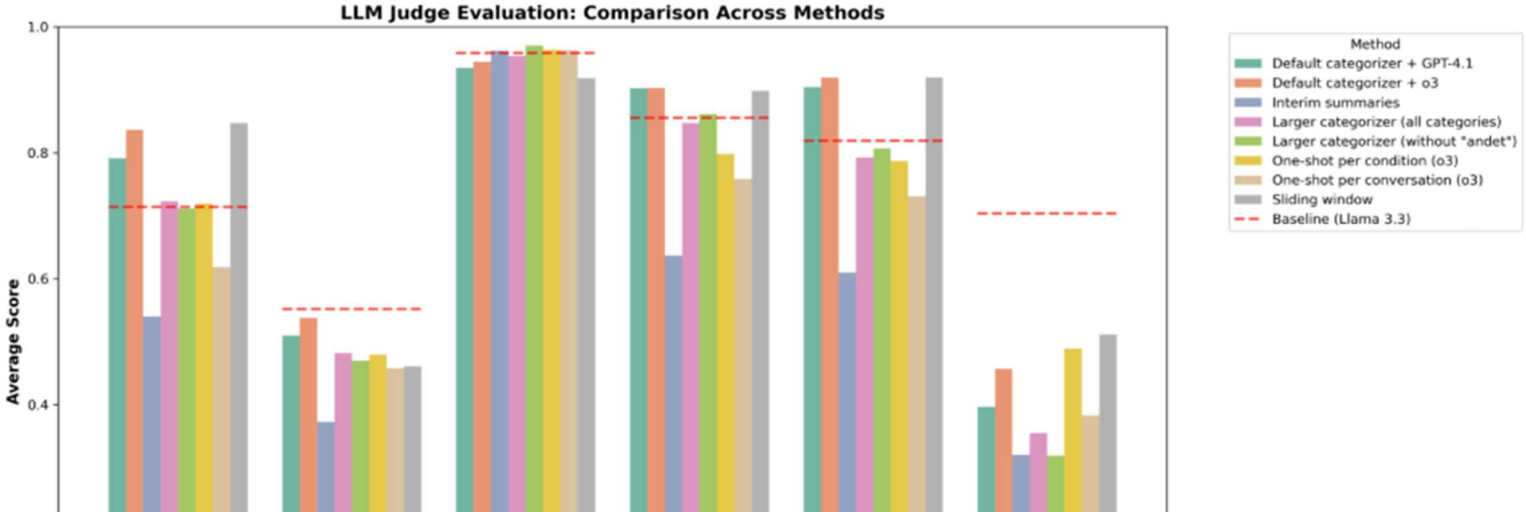
(0.73) og metode 5.1 (0.72). Dette resultat skal dog tolkes med forsigtighed, da evalueringsgrundlaget (de manuelt korrigerede relevansmarkeringer) bygger på afprøvningssetuppets output, hvilket kan introducere bias, der favoriserer afprøvningssetuppet kunstigt.

Tabel 3: Relevansklassificering

Metode	Accuracy	Precision	Recall	F1-Score
Afprøvningssetup (baseline)	**0.88**	**0.81**	0.78	**0.79**
Metode 1 (Delopsummeringer)	0.84	0.76	0.72	0.74
Metode 2.1 (One-Shot per tilstand)	0.73	0.53	**0.88**	0.66
Metode 2.2 (One-Shot per conversation)	0.77	0.58	0.84	0.68
Metode 3.1 (Alt. kategorisering)	0.82	0.75	0.59	0.66
Metode 3.2 (Alt. kategorisering)	0.81	0.65	0.80	0.72
Metode 4 (LLM Sliding Window)	0.76	0.69	0.26	0.38
Metode 5.1 (GPT-4.1)	0.82	0.69	0.76	0.72
Metode 5.2 (o3)	0.84	0.76	0.71	0.73

På kvalitetsvurdering, hvor GPT-4o evaluerer notaterne på seks dimensioner, excellerer de større modeller markant på sproglige parametre sammenlignet med afprøvningssetuppet. Metode 5.2 opnår højest *coherence* (0.84 mod 0.71), *professionalism* (0.92 mod 0.82) og *grammatical correctness* (0.90 mod 0.86). Til gengæld er afprøvningssetuppet mest koncis (0.70 mod 0.40-0.51). Dette kunne indikere at de større modeller genererer mere sammenhængende og udførlige notater på bekostning af de blive længere og mindre koncise. På *faithfulness* scorer alle metoder højt (0.92-0.97), hvilket viser, at information fra samtalen bevares korrekt, selvom *correctness* er lavere for alle alternative metoder. Dette indikerer, at de alternative metoder gengiver den samme information fra samtalerne, men formulerer den anderledes end afprøvningssetuppets validerede notater - den lavere *concision* skyldes altså mere udførlig gengivelse snarere end hallucination eller tilføjet irrelevant information. Se figur 6 for et fuldt overblik over 'LLM-as-a-judge'-evalueringen.

Figur 8: 'LLM-as-a-judge' kvalitetsvurdering



Praktiske implikationer

Det primære forskningsspørgsmål for denne analyse var: Kan vi forbedre kvaliteten af automatisk genererede sygeplejetilstandsnotater ved at udforske forskellige genereringsmetoder uden begrænsninger på hardware eller eksekveringstid? Svaret er ja – men kvalitetsforbedringerne fordeler sig forskelligt på tværs af evalueringsparametre, hvilket nødvendiggør bevidste valg baseret på anvendelseskontekst.

Hvis det er kritisk at minimere både falske positiver (irrelevante tilstande der genererer unødvendig dokumentation) og falske negativer (væsentlige tilstande der overses), fremstår metode 1 som det stærkeste valg med F1=0.74. Dette skal dog afvejes mod metodens markant lavere kvalitet af sygeplejetilstandsnotater (overall average 0.57), hvilket indebærer, at de genererede notater vil kræve betydelig manuel revision.

Omvendt, hvis ressourcer tillader anvendelse af state-of-the-art modeller, viser metode 5.2, den bedste balance med både høj overall kvalitetsscore (0.77 - samme niveau som afprøvningssetuppet) og rimelig relevansvurdering (F1=0.73). Metoden excellerer særligt på sproglige dimensioner, hvilket resulterer i notater der kræver mindre sproglig redigering. Metode 5.1 viser lignende styrker (kvalitetsscore 0.74, F1-score 0.72) og kan være et omkostningseffektivt alternativ.

Det skal bemærkes, at evalueringens største begrænsning er, at ground truth bygger på afprøvningssetuppets output, hvilket introducerer bias. Desuden mangler evalueringen klinisk validering af de genererede notater. Før implementering i produktionssammenhæng bør valgte metoder derfor valideres med uafhængig ground truth og klinisk ekspertise for at sikre både faglig rigtighed og praktisk anvendelighed. For en mere dybdegående gennemgang af analysearbejdet på Use case 1 se supplerende dokument: "Analyser"

2.3 Evaluering

Projektet besluttede tidligt i processen at vi skulle udstille og afprøve vores løsninger til repræsentanter fra en voksende gruppe afprøvningskommuner. Dette satte en række krav til vores løsninger, men betød også at vi havde en oplagt metode til at indsamle evalueringsdata og måle performance på vores løsninger. Foruden denne evaluering blev metrikker også etableret under udvikling for at styre udviklingen hen imod en løsning af tilstrækkelig kvalitet til at understøtte en afprøvning.

Under udvikling optimeres de enkelte komponenter i løsningen med henblik på at opnå en tilstrækkelig samlet performance til understøttelse af afprøvning. Dette var primært relevant op imod løsningens første afprøvning, da vi herfra kunne overgå til at evaluere performance på data indsamlet via afprøvning og efterfølgende behandling. For yderligere information om data indsamlet til evaluering under udvikling og på baggrund af afprøvninger se supplerende dokument: "Data og modeller vi har"

Evaluering under udvikling

Under udvikling optimeres de enkelte komponenter i løsningen med henblik på at opnå en tilstrækkelig samlet performance til understøttelse af afprøvning. Dette var primært relevant op imod løsningens første afprøvning, da vi herfra kunne overgå til at evaluere performance på data indsamlet via afprøvning og efterfølgende berigelse.

Til evaluering under udvikling blev en baseline etableret, opgjort af et datasæt og et tilhørende sæt af metrikker for hver komponent. Datasættet består i sin første version af 12 lydoptagelser af manuelt syntetiske SFU-samtaler med tilhørende journaltekst. Samtalernes transskriptioner er rettet og dialoglinjerne er kategoriseret. Udredning af baselinen gav anledning til forskellige brug og konklusioner for de forskellige komponenter.

Talegenkendelse (TGK)

Baseline datasættets optagelser og tilhørende rettede transskriptioner blev brugt til at udregne word error rate (WER) og character error rate (CER). Dette blev benyttet til at sammenligne forskellige talegenkendelsesmodeller og konfigurationer af samme model. Det blev også brugt til at vurdere behovet for videreudvikling på TGK-komponenten i kontekst af SFU.

Foruden evaluering af performance, blev der også løbende foretaget hastighedsmålinger for TGK-komponenten, både for den enkelte model for enkelte samtaler og for serverens performance ved større load.

Kategorisering

Baseline datasættes dialoglinjer og tilhørende kategorier blev brugt til at evaluere kategoriseringsmodellens performance. En række metrikker indgik i denne evaluering: balanced_accuracy[2], accuracy[3], f1-score[4], precision[5], and recall[6].

Disse metrikker informerer valg af modelarkitektur og endeligt, hvilken model der bruges til afprøvning. Vores primære metrik er balanced accuracy, da SFU-samtaler indeholder en betydelig mængde snak der ikke informerer journalteksten for en bestemt sygeplejetilstand, hvorfor det er kategoriseret under en 'andet'-kategori. Vi ønsker at vægte sygeplejetilstandene højere end 'andet', hvorfor balanced accuracy er en mere robust måling.

Opsummering

Den indledende prompt til opsummering blev udviklet med baseline datasættet og dets forventede journaltekst. Videreudvikling på opsummeringskomponenten har lænet sig op ad data indsamlet fra afprøvningerne. Hvor prompten løbende er blevet tunet fra afprøvning til afprøvning med henblik på at ramme flere relevante opsummeringer, og lave færre fejl i de opsummeringer, hvor vi har modtaget rettelser. Denne proces er altså styret af en kvalitativ vurdering, hvor det forventede journaltekst genereret af fagligt personale sammenholdes med udkast fra modellen.

Vi har senere i projektet arbejdet med etablering af mere kvantitative metrikker til evaluering af modellens udkast til journaltekst. Derudover har vi eksperimenteret med at bruge LLM'er til automatisk

vurdering af kvalitet – denne proces kaldes 'LLM-as-a-judge'. Førnævnte er gjort som mindre analyser på bagkant af afprøvningerne og dokumentation om dette vil blive offentliggjort senere.

Evaluering af afprøvninger

I forbindelse med afprøvning af vores løsning til understøttelse af SFU-samtaler optager og efterbehandler sygeplejersker fra projektkommunerne samtaler i løsningen. Dernæst behandles den konkrete samtale ved at løsningens udkast til journalnotater (opsummeringer) sammenholdes med den komplette liste af sygeplejetilstande der på baggrund af samtalen bør oprettes notater til, vi refererer til denne liste som de **relevante** sygeplejetilstande. Dernæst tager sygeplejersken stilling til hvorvidt den enkelte opsummering kræver rettelser for at være fyldestgørende. Afslutningsvis markeres irrelevante sygeplejetilstande således.

Sygeplejersken tager altså, for hver sygeplejetilstand, stilling til følgende:

1. Hvorvidt sygeplejetilstanden er relevant for samtalen
2. Hvorvidt løsningens opsummering kræver rettelser
3. Hvorvidt relevante sygeplejetilstande manglede et udkast til notat og derfor skulle oprettes manuelt

En samlet afprøvning og tilhørende efterbehandling generer en større samling SFU-samtaler. Denne data benyttede vi på bagkant til at evaluere performance på vores løsning. Performance blev sammenlignet på tværs af afprøvninger ved etablering af en samling af metrikker, en baseline.

De vigtigste metrikker for baselinen var følgende metrikker:

a. Andelen af alle **relevante** opsummeringer løsningen leverer
b. Andelen af løsningens relevante opsummeringer der ikke krævede rettelser
c. Andelen af løsningens relevante opsummeringer der krævede rettelser
d. Andelen af alle **irrelevante** opsummeringer løsningen leverer

På bagkant af en afprøvning fortæller den samlede baseline om i hvor står grad løsningen leverede de opsummeringer der var forventet (a), hvor ofte de skulle rettes (b, c), og hvor meget løsningen støjede (d).

Baselinen er efter hver afprøvning blevet suppleret af en analyse af, hvilke(n) komponent i løsningen der gav anledning til en given fejl. Dette er primært gjort med fokus på opsummeringer med rettelser (c). Analysen udføres ved at kigge på transskriberingen til samtalen, teksten der blev klassificeret til den pågældende sygeplejetilstand, og den endelige opsummering. Her blev det for hver opsummering noteret, hvor fejlen opstod i systemet og hvad fejlen var. Den information giver et detaljeret indblik i hvor i løsningens komponenter der er mulighed for forbedring.

De samlede resultater fra afprøvningerne på UC1 kan ses i tabel 5. På grund af de stigende antal brugere, havde anden afprøvning et større antal samtaler. Ved første afprøvning blev 30 SFU-samtaler optaget, hvoraf 231 sygeplejetilstande var relevante. Dette antal steg ved anden afprøvning til 54 SFU-samtaler, med 378 relevante sygeplejetilstande.

Tabel 4: Resultater fra UC1 afprøvninger

Afprøvning	1	2
Antal relevante ops.	231	378

Andel relevante ops. identificeret		80,1 % (185)	78,3 % (296)
Andel relevante ops. udkast godkendt	u/rettelser	21,2 % (49)	30,7 % (116)
	m/rettelser	58,9 % (136)	47,6 % (180)
Andel relevante ops. misset		19,9 % (46)	21,7 % (82)
Antal ikke relevante ops.		459	864
Andel ikke relevante ops. identificeret		90,0 % (413)	91,8 % (793)

Efter hver afprøvning blev data analyseret med henblik på at identificere og implementere forbedringsmuligheder. Ved anden afprøvning missede systemet en relevant opsummering 21,7% (82) af de 378 relevante sygeplejetilstande. Analysen af, hvor i systemet fejlene stammer fra, viste følgende resultater:

- Talegenkendelse: Årsagen til fejl i 30% af tilfældene
- Kategorisering: Årsagen til fejl i 29% af tilfældene
- Opsummering: Årsagen til fejl i 41% af tilfældene

Nedenfor i tabel 6 er der lavet en oversigt over fordelingen af de typer af fejl hver komponent laver ud af alle fejl. Det ses her, at opsummeringen har en forholdsvis jævn fordeling af fejltyper mens talegenkendelsen er domineret af stavefejl og kategoriseringen hovedsageligt er årsag til manglende informationer.

Tabel 5: Oversigt over fejltyper UC1 afprøvning 2

Komponent \ Fejltype	Mangler information	Indsat irrelevant information	Indsat forkert information	Stavefejl	Dårligt / upræcist sprog
Talegenkendelse	9%	2%	2%	19%	-
Kategorisering	22%	3%	2%	-	-
Opsummering	9%	9%	8%	3%	13%

Fordelingen af komponent og type af fejl under afprøvning. Værdierne er angivet som andelen af alle fejl. Bemærk at mængden af afrundinger har gjort, at der både summeres til 101% og at de ikke stemmer helt overens med de overordnede procentsatser for hver komponent.

2.4 Videreudvikling

Det primære behov der er nødt til at blive adresseret for løsningen i kontekst af videreudvikling er mistanken om kvaliteten af det indsamlede evalueringsdata. Vi har en kraftig mistanke om at den behandling der er foretaget af samtalerne ifm. afprøvningerne, er påvirket af løsningens output i sådanne grad at vi har en begrænset evne til vurdere, hvorvidt præcisionen og kvaliteten af løsningens output kan forventes at translatere til virkeligheden. En mindre biased måling bør blive udført, før behovet for videreudvikling endeligt kan klargøres.

Ud over en opdateret vurdering af den aktuelle værdiskabelse, ser vi fra teknisk perspektiv et behov for at bygge løsningen op omkring en iterativ udvikling, hvor den måde løsningen skaber værdi på, gradvist bliver mere og mere kompleks, men tager udgangspunkt i et simplere løsningsdesign end det nuværende.

Supplerende til det, ser vi i projektet, et behov for at løsningen mere fyldestgørende understøtter den lange liste af aktiviteter og dokumentationsopgaver der er forbundet med en SFU-samtale.

Sidste, men ikke mindst, ser vi selvfølgelig et potentiale for forbedring af løsningens komponenter.

Alle disse punkter adresseres i dette afsnit i form af uddybende forklaringer og anbefalinger til videreudvikling.

Evalueringsdata

Da projektet, som beskrevet ovenfor, har haft begrænset mulighed for at måle den potentielle værdiskabelse af løsningen, må et bedre evalueringssæt indsamles. Det kan gøres på flere forskellige måder, men hver deres potentielle faldgruber, begrænsninger og gevinster.

- Med udgangspunkt i det nuværende evalueringssæt

For at den ønskede måling skal kunne udføres på det nuværende evalueringssæt skal sættes efterbehandles. Denne efterbehandling er potentielt ret indgribende og har til formål at udbedre fejlbehandlede notater, relevansmarkeringer og potentielt hele samtaler. Denne tilgang sikrer et relativt stort evalueringssæt uden store omkostninger forbundet med indsamlingen, men risikerer at have begrænset effekt, da datasættet er begrænset af at være funderet i syntetiske samtaler og at det forventeligt ikke er muligt at fjerne den omtalte bias helt.

- Med udgangspunkt i ægte data

En anden tilgang er at genskabe et helt nyt evalueringssæt baseret på en repræsentativ samling af ægte samtaler. Hvor rigtige sygeplejefaglige udredningssamtaler med borgere optages og tilhørende journalnotater noteres og kvalitetssikres. Sammenlignet med et evalueringssæt baseret på syntetiske samtaler, har denne tilgang langt større sandsynlighed for at sikre en måling, der kan forventes at translaterer til virkeligheden. Tilgangen er til gengæld langt mere omkostningsfuld og hvis tilstrækkelig kvalitetssikring ikke udføres, kan målingen blive begrænset eksisterende dokumentationspraksis og ikke den ideelle praksis.

Fælles for alle evalueringssæt og deres tilhørende kvalitetsmåling vil være behov for variation. For at et evalueringssæt kan benyttes til en fyldestgørende besvarelse af spørgsmål: “Kan løsningen, på baggrund af en optaget SFU-samtale, generere værdiskabende journalnotater?” kræver det at datasættet opgjort af en samling SFU-samtaler der er repræsentative på et stort udvalg af parametre. Værdiskabelse målt på et datasæt der er repræsentativt for SFU-samtaler og tilhørende journalnotater i Københavns Kommune vil højst sandsynligt ikke translaterer uden tab til Aarhus Kommune.

Variation i evalueringssættet

For at evalueringssættet er repræsentativt skal tilstrækkelig variation i følgende parametre være repræsenteret:

- Lyd
  - Baggrundsstøj
  - Lydstyrke på talere
- Stemme
  - Sprogbrug
  - Dialekt
  - Accent
  - Tempo
- Indhold
  - Helbredsudfordringer
  - Opfølgningsmuligheder
  - Kompleksitet

Store dele af overordnede sygeplejefaglige og indholdsmæssige parametre forventes at være relativt konstante på tværs af landet, hvorfor de lokale sproglige og potentielt lydmæssigt variationer bør være i fokus i den brede dataindsamling. Forståelse af omfanget af data nødvendigt for at dække en repræsentativ mængde variation i ovenstående parametre er centralt for kunne indsamle et evalueringssæt.

Iterativ værdiskabelse

Simplificeret har den nuværende løsning har tre centrale opgaver: 1. transskriber samtalen, 2. identificer hvilke emner der er afdækket i samtalen og 3. generer relevante sygeplejetilstandsnotater på baggrund af samtalens indhold.

Vi valgte i projektet at udvikle og afprøve en løsning der understøttede alle 3 opgaver, ud fra et perspektiv er at den centrale værdiskabelse ligger i tidsbesparelse forbundet med generering af brugbar journaltekst.

Hvis sådanne tidsbesparelse skal realiseres, skal løsningens nuværende notatspræcision og -kvalitet forbedres betydeligt. Det er potentielt en omkostningsfuld proces, med en reel sandsynlighed for at løsningen ikke opnår tilstrækkelig kvalitet.

Derfor bør man kraftigt overveje om andre nærtliggende løsningsdesigns kan skabe tilstrækkelig værdi til at retfærdiggøre idriftsættelse og implementering i sygeplejefaglige praksis. Et konkret forslag til en sådanne løsning er et løsningsdesign der understøtter følgende tre centrale opgaver: 1. transskriber samtalen, 2. identificer hvilke emner der er afdækket i samtalen, 3. generer opsummering af emner på baggrund af samtalens indhold.

En sådanne løsning skaber ikke værdi med at genere notaterne, men i stedet ved at støtte journalskrivningen. Denne løsning har den primære fordel at have en væsentlig lavere barriere for at indfri sit potentiale, samtidig kan den implementeres på en måde der skaber ideelle forhold for at udvikle den ideelle løsning på sigt, da forskellen på præcision og kvalitet af en opsummering af samtalen og et fremtidigt journalnotat isoleres til den nødvendige kontekst til at formulere notatet.

Udvidet understøttelse af SFU-samtalen

En oplagt mulighed for videreudvikling på løsningen er en udvidet understøttelse af SFU-samtalen. Oprettelse og vedligehold af sygeplejetilstandene er en central del af SFU-samtalen, men sygeplejerskerne foretager en del andre oprettelser og opdateringer i borgerens journal på baggrund af samtalens indhold. Denne praksis forventes at være forskellig på tværs af kommunerne, men der foregår formegentlig oprettelse eller opdatering af centrale felter i borgerens journal, oprettelser der

vil være lignende på tværs af landet. Udvidelser af understøttelsen kan passende implementeres i tråd med ovenstående budskab om iterativ værdiskabelse.

Komponentforbedringer

På bagkant af en genetablering af evalueringssættet kan de enkelte komponenter i løsningen forbedres på flere parametre. Forhåbentlig vil analyserne gennemgået længere oppe belyse et udvalg af forbedringsmuligheder, hvis de genkøres på det nye evalueringssæt.

Ud over de oplagte forbedringsmuligheder: Forbedring af talegenkendelses- og kategoriseringskomponenten via implementering af modeller med bedre præcision og yderligere promptforbedringer til opsummeringskomponenten.

Er der en relateret men separat forbedringsmulighed for promptforbedringer gennem yderligere kontekst fra borgerjournalen. Borgerbesøget udføres aldrig i isolation, hvorfor relevant kontekst til besøget bør være tilgængelig i borgerjournaler til relevansvurdering og generering af journalnotater.

2.4 Idriftsættelse

UC1 er på nuværende tidspunkt ikke klar til idriftsættelse i et produktionsmiljø. Dette afsnit beskriver derfor perspektiver og centrale overvejelser ved en fremtidig idriftsættelse, baseret på erfaringer fra afprøvninger under projektperioden samt input fra teknisk spor.

En eventuel idriftsættelse vil afhænge af en række faktorer, herunder den endelige tekniske løsning, valg af integrationsmodel (integration i EOJ-system vs. ekstern service), og afklaring af forretningsmæssige krav. De følgende afsnit behandler de væsentligste områder, der skal adresseres i forbindelse med en fremtidig idriftsættelse.

Dataflow og integrationer

Under afprøvningen af UC1 blev der anvendt en mock-up løsning uden reelle integrationer til eksisterende EOJ-systemer. Dette afsnit beskriver nogle af de forventede integrationsbehov, der vil være relevante ved en eventuel idriftsættelse. Som løsningen udvides, forventes det også, at integrationsbehovene udvides.

Integrationsscenarier

Der er grundlæggende to mulige scenarier for integration af en AI-løsning til sygeplejefaglig udredning:

Eksternt tredjepartssystem: I dette scenarie fungerer AI-løsningen som et separat system, som sundhedspersonalet skal skifte til fra deres EOJ-system. Erfaringer fra sundhedssektoren viser, at brugere generelt ikke ønsker at skifte mellem systemer under arbejdsgange, hvilket gør denne løsning mindre attraktiv.

Integration i EOJ-system (anbefalet): I dette scenarie integreres AI-funktionaliteten direkte i EOJ-systemet. Brugeren aktiverer funktionen fra EOJ-systemets brugergrænseflade, hvorefter systemet kommunikerer med AI-komponenten via et API, og resultatet returneres direkte til EOJ-systemet. Denne tilgang understøtter en mere smidig arbejdsgang og mindsker risikoen for at brugerne ikke anvender løsningen.

Relevante integrationer

Ved integration i EOJ-system vil følgende integrationer være relevante at overveje:

Data-integration: For at optimere kvaliteten af den genererede dokumentation vil der være behov for at AI-løsningen kan bruge udvalgte borgerdata fra EOJ-systemet. En del af at udvikle en brugbar løsning vil være at undersøge hvilke data der er nødvendigt at bruge for at give et godt resultat.

Journaliseringsintegration: Når AI-løsningen har genereret et udkast til dokumentation, vil det være hensigtsmæssigt at dette kan gemmes direkte i borgerens journal i EOJ-systemet. Dette kræver API-adgang til journaliseringsfunktionaliteten.

Sikkerhed

Sikkerhed er en central overvejelse ved idriftsættelse af en AI-løsning til håndtering af følsomme sundhedsdata. Sikkerhedsløsningen vil dog afhænge af, hvordan løsningen udbydes og implementeres.

Integreret i EOJ-system

Hvis AI-løsningen integreres direkte i EOJ-systemet, vil sikkerheden primært håndteres gennem EOJ-systemets eksisterende sikkerhedsmekanismer. I dette scenarie er sikkerhed allerede en del af den grundlæggende infrastruktur, som leverandøren af EOJ-systemet leverer.

Ekstern service

Hvis AI-løsningen leveres som en ekstern service, vil der være behov for separate sikkerhedsforanstaltninger, herunder sikker kommunikation mellem EOJ-systemet og AI-tjenesten. I dette scenarie er det vigtigt at sikre, at AI-tjenesten lever op til de samme sikkerhedsstandarder som EOJ-systemet, herunder krav til databeskyttelse og GDPR-compliance.

Ressourcebehov og skalerbarhed

Der er ikke foretaget formelle performance-målinger under projektperioden, men erfaringerne fra afprøvningerne giver indikationer om fremtidige skaleringsudfordringer.

Observationer fra afprøvninger

Under afprøvninger med op til 8-12 samtidige brugere blev der observeret performance-udfordringer. Dette indikerer, at den nuværende implementering af løsningen vil kræve betydeligt arkitekturarbejde for at kunne håndtere en produktionssituation med mange samtidige brugere. Som det blev bemærket under projektperioden: løsningen skaler ikke tilstrækkeligt i sin nuværende form.

Potentielle skaleringsløsninger

For at adressere skaleringsudfordringerne vil der være behov for arkitekturarbejde. Mulige tilgange til forbedret skalering kunne omfatte:

Intelligent batching: Optimering af hvordan anmodninger grupperes og processeres kan forbedre udnyttelsen af tilgængelige ressourcer.

Dedikeret infrastruktur: Opdeling af systemet, så forskellige komponenter (talegenkendelse, klassificering, og opsummering) kører på dedikeret infrastruktur, kan forbedre performance og skalering.

Model-optimering: Det kan blive nødvendigt at træne flere eller optimerede modeller for at opnå den nødvendige performance ved højere belastning.

Det skal dog understreges, at disse løsninger ikke er undersøgt i detaljer under projektperioden, og en konkret vurdering af hvilke tilgange der er mest effektive vil kræve yderligere analyse og test.

Miljøopsætning

Miljøopsætningen for en produktionsklar løsning vil afhænge af den endelige tekniske løsning og de forretningsmæssige krav. Som det blev fremhævet under projektperioden, er dette et af de vigtigste områder at forholde sig til ved idriftsættelse.

Logging og metrics

En forudsætning for at kunne dimensionere den nødvendige infrastruktur er implementering af systematisk logging. Dette omfatter:

- Logging af responstid for hvert kald til talegenkendelsesmodellen
- Logging af responstid for hvert kald til klassificeringsmodellen
- Logging af responstid for hvert kald til opsummeringskomponenten
- Registrering af token-forbrug per kald

Med disse data vil det være muligt at analysere systemets adfærd under forskellige belastningsniveauer og beregne det faktiske hardware-behov. Denne type logging var ikke implementeret i afprøvningsløsningen, men vil være nødvendig for en produktionsløsning.

Afhængighed af teknisk design

Desuden afhænger miljøopsætning og hardware-behov i høj grad af de tekniske designvalg, der træffes. Erfaringer fra lignende AI-løsninger viser, at selv tilsyneladende små forskelle i implementering kan have betydelig indflydelse på performance. Valg af prompt-strategier, hvordan data processeres, og hvordan forskellige komponenter integreres kan alle påvirke den samlede ressourceudnyttelse og responstid væsentligt.

Dette illustrerer, at det ikke kun er mængden af hardware, men også hvordan systemet er designet, der afgør den endelige performance. Derudover vil forretningsmæssige krav spille ind. Hvis løsningen skal levere resultater i realtid, stiller det andre krav til infrastrukturen end hvis batch-processering om natten er acceptabelt. Disse faktorer skal afklares før en endelig miljøopsætning kan specificeres.

Tidslinje

En konkret tidslinje for idriftsættelse af en produktionsklar løsning vil afhænge af afklaring af en række fundamentale spørgsmål. Før en realistisk tidsplan kan udarbejdes, er det nødvendigt at træffe beslutninger om følgende områder:

- Klinisk/faglig afklaring af funktionalitet:

I det implementerede proof of concept blev løsningen afgrænset til kun at skulle kunne give et forslag til dokumentation for "Beskrivelse af tilstand". Dette er kun en mindre del af det, der skal dokumenteres. Løsningen forholder sig desuden ikke til allerede eksisterende data. Der er derfor et behov for at lave en klinisk/faglig gennemgang af, hvad det reelle behov i løsningen er i forhold til funktionalitet. Først når dette er afklaret vil man reelt kunne gå i gang med at designe og fastlægge en tidslinje.

- Metrikker og kvalitetskrav:

Hvilke metrikker skal løsningen måles på, og hvad er de acceptable kvalitetsniveauer? Dette inkluderer både acceptable og uacceptable fejlrater samt krav til nøjagtighed af den genererede dokumentation.

- Arkitektur-design:

Den endelige arkitektur skal designes med henblik på de identificerede skaleringsudfordringer. Dette inkluderer beslutninger om hvordan forskellige komponenter integreres og distribueres.

- Forventet belastning:

Hvor ofte skal løsningen anvendes? Skal den håndtere tusindvis af kald dagligt, eller er det mere begrænset brug? Dette har direkte betydning for dimensionering af infrastruktur.

- Performance-krav:

Hvad er maksimalt acceptable responstid for løsningen? Skal den fungere i realtid, eller er batch-processering acceptabelt?

- Ressourcer og hardware:

Baseret på ovenstående skal det konkrete hardware-behov fastlægges, hvilket kræver de logging-mekanismer beskrevet under miljøopsætning.

Når disse spørgsmål er afklaret, vil det være muligt at lave en mere præcis vurdering af implementeringstid og ressourcebehov for idriftsættelse.

3. Journalopsummering (UC2)

3.1 Formål og beskrivelse

Use casen omhandlende journalopsummeringer har et overordnet formål om at nedbringe forberedelsestiden og øge kvaliteten af den forberedelse, som sygeplejefagligt personale får inden et besøg hos en borger. Dette skal ske gennem genererede opsummeringer af borgerjournaler, så personalet derved slipper for at bruge tid på at lede i borgernes journaler efter viden inden et besøg, men at de i stedet kan få den viden de skal bruge til at udføre besøget leveret i en samlet opsummering.

Journalopsummeringen er opdelt i to separate løsninger, der henvender sig til hver sin faggruppe; hhv. udekørende sygeplejersker og udekørende SSH/SSA. Sygeplejerskerne foretager sundhedsfaglige vurderinger og undersøgelser af borgeren, mens SSH/SSA primært tager sig af den praktiske hjælp og personlige pleje hos borgeren. Det er denne forskel i arbejdsopgaver, der giver forskellige behov i forbindelse med forberedelsen af deres besøg og dermed også af indholdet i en journalopsummering.

Intentionen med løsningen for sygeplejerskerne var at køre det iterativt, ved først at undersøge, hvordan vi kan lave meningsfulde generelle opsummeringer af journaldata, som i sig selv vil henvende sig mest til nye medarbejdere, vikarer og meget generelle besøgskontekster, hvor der kan være behov for at modtage en bred og generel opsummering af borgerens journal. Næste skridt skulle dernæst have været at bygge ovenpå ved at lave besøgsspecifikke opsummeringer, der tilpasser opsummeringen til præcis den type af besøg som sygeplejersken er ude og udføre. Dette blev dog ikke nået i dette projekt. For SSH/SSA er deres besøgsplan så central en del af deres besøg og dermed også af deres forberedelse, at det ikke gav mening at udelade den i første omgang. Derfor blev deres opsummeringer fra starten af specifikke for besøget da besøgsplanerne indgik i opsummeringen, men delene omhandlende borgerens sygeplejetilstand og anden vigtig information udover besøgsplanen blev stadig opsummeret på et generelt niveau, hvilket er vurderet tilstrækkeligt for denne faggruppes besøg, og derfor ikke kræver yderligere udvidelser.

Afgrænsninger

Løsningen retter sig mod sygeplejefagligt personale, der skal foretage besøg hos en ukendt borger. Med ukendt borger, menes der her en borger, som personalet enten ikke har en daglig kontakt med eller hvor der er store skift i, hvilket personale der kommer ud til borgeren. Det henvender sig dermed bl.a. mod det udekørende personale, nyt personale og vikarer, og omvendt henvender det sig ikke til fastansat personale på f.eks. plejecentre. Her vil de kende deres borgere mere gennemgående, og derfor ikke have behov for en større opsummering af borgerens journal, men i større grad have behov for, mindre opdateringer på, hvad der er sket siden sidste besøg.

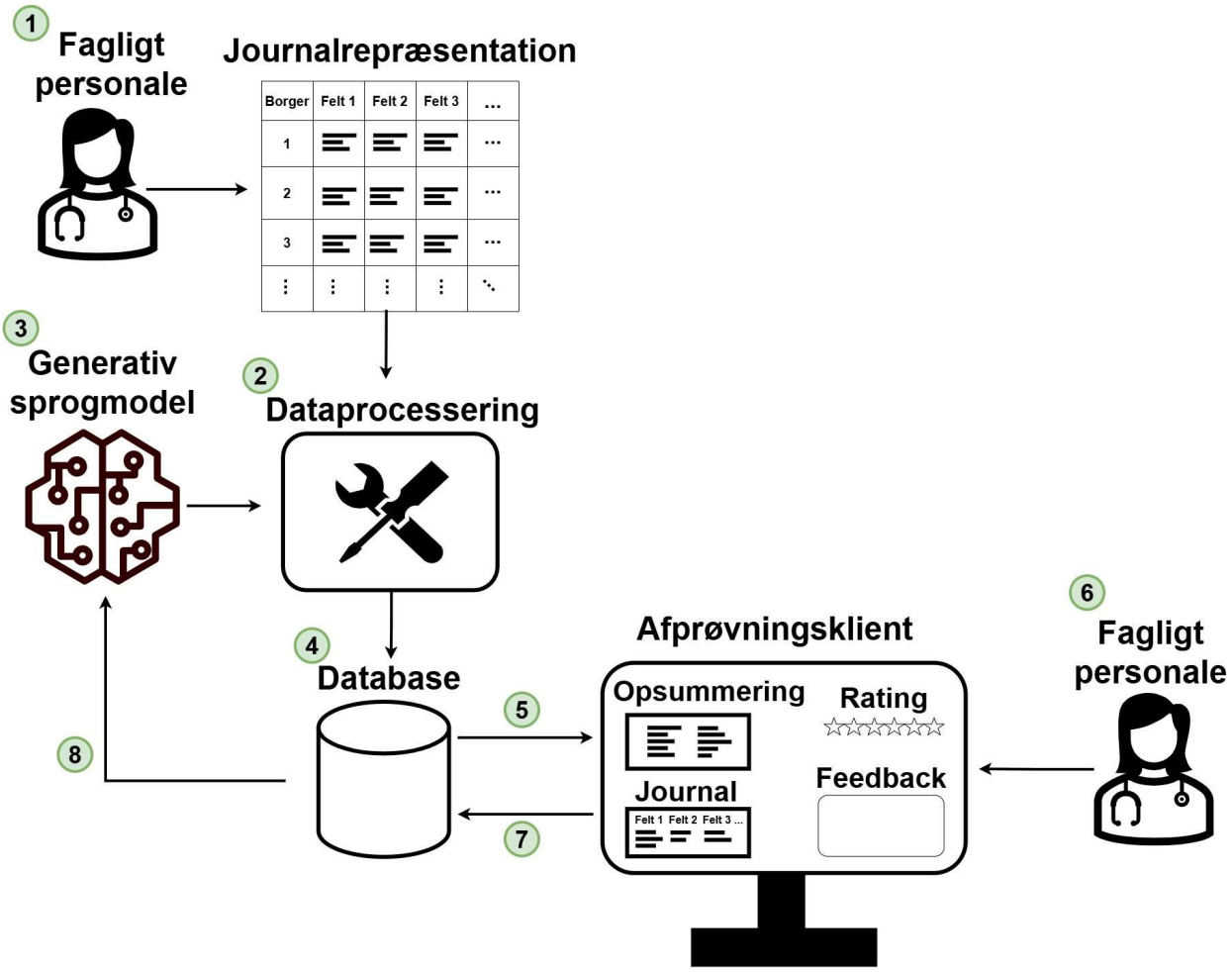
Data i løsningen er udvalgt af fagligt personale ud fra en vurdering af, hvad der var de vigtigste oplysninger for personalet i forbindelse med et besøg, og de to løsninger indeholder derfor en mindre afgrænset datamængde. Løsningen afspejler derfor ikke et realistisk scenarie, da hverken kompleksiteten ift. datamængden eller selve udtrækningen af data fra EOJ-systemet er afdækket.

3.2 Teknisk løsning

Løsningen bygger på en repræsentation af journaldata med tilhørende fagligt-genererede opsummeringer af journalerne. Data er blevet struktureret i et skema, som har muliggjort en struktureret behandling af data. Komponenterne og flowet for løsningen er illustreret i figur 7, og indeholder følgende punkter:

1. Det faglige personale indskriver journalfelter for hver borger i et digitalt skema.
2. Data processeres ved at udtrække data fra journalpræsentationen og strukturere det således at data kan indhentes og indsættes korrekt i prompten af sprogmodellen.
3. Den generative sprogmodel genererer en journalopsummering baseret på journaldataen. Prompten tunes initialt med henblik på at generere opsummeringer, der ligner de manuelt genererede opsummeringer.
4. Opsummering og tilhørende journaldata indskrives i databasen
5. Opsummering og tilhørende journaldata indlæses af afprøvningsklienten.
6. Fagligt personale tilgår afprøvningsklienten og giver opsummeringerne en rating og feedback-kommentar.
7. Rating og feedback journaliseres i databasen
8. Analyse af rating og feedback fra afprøvning giver anledning til opdateringer i sprogmodellens prompt, og nye opsummeringer genereres, der på ny kan blive afprøvet.

Figur 9: Journalopsummering – Teknisk løsningsopbygning



Nedenfor er der en yderligere uddybning af, hvordan afprøvningsklienten, dataprocesseringen og prompt-strukturen for sprogmodellen er opbygget.

Afprøvningsklient

Afprøvningsklienten er udviklet som et evalueringsværktøj til fagpersonale, der skal teste kvaliteten af systemets genererede opsummeringer af journaldata. I de følgende afsnit beskrives klientens brugergrænseflade, funktionalitet og de designovervejelser, der ligger til grund for løsningen.

Brugergrænseflade

Afprøvningsklienten er bygget til at understøtte testning af Use casen, hvis primære formål er at gøre det muligt for fagpersonale at evaluere kvaliteten af de genererede opsummeringer.

Fagpersonalet skal vurdere opsummeringerne ved at læse det tilhørende journaldata og dernæst give deres feedback. Både journaldata og opsummeringerne er forberedt på forhånd, så fagpersonalet skal ikke selv oprette eller generere data. Datagrundlaget er udarbejdet i samarbejde med fagpersonale forud for afprøvningen.

Klienten indeholder følgende information:

- For begge Use-cases
  - Generelt journaldata på borgeren
  - Feedback-modul til evaluering af opsummeringer
- SPL-Use case
  - Én opsummering per borger
  - Tilhørende besøgskontekst
- SSH/SSA-Use case
  - Én opsummering per besøgstidspunkt (morgen, middag, aften og nat)
  - Besøgsbeskrivelse for det pågældende besøgstidspunkt

Klientbeskrivelse

Brugergrænsefladen viser en række journaler, som der skal gives feedback på jf. figur 8a. Navnet på borgeren vises samt en mulig besøgskontekst, der kort beskriver, hvad fagpersonalet skal forholde sig til, når de læser journalen. For hver borger i journaloversigten er der en indikator, som viser, om brugeren har givet feedback på den givne journal. Dette vises med henholdsvis grønt (feedback afgivet) og rødt (feedback afventer).

Når man trykker på borgeren i SPL-Use casen, så vil man blive navigeret ind på journalen, hvor der præsenteres: en besøgskontekst, en opsummering af journaldata, den fulde journal og et feedbackmodul jf. figur 8b. Den fulde journal kan foldes ud, jf. figur 8c. Journalens sektioner består af: Generel information, sygeplejetilstande og observationer. Disse sektioner er udvalgt af fagpersonale for at afgrænse Use casen.

For at evaluere opsummeringen af journaldata, skal brugeren læse opsummeringen og kigge i journaldataet for at se, om opsummeringen er fyldestgørende og beskrivende nok for de rammer, der er givet i besøgskonteksten. I feedbackmodulet nederst kan brugeren give opsummeringen en score fra 1-6 stjerner, og hvor 6 er bedst. Det primære feedback kommer i form af den tekst, som brugeren skriver i et fritekstfelt. Derfor kan man først sende feedback, hvis man udfylder en beskrivelse af, hvorfor opsummeringen er god/dårlig, eller hvad den mangler.

I SSH/SSA-Use casen gør brugergrænsefladen brug af mange af de samme komponenter. Forskellen ligger i, at besøgskonteksten er givet i form af besøgstidspunktet og besøgsbeskrivelsen. Ved afprøvning starter brugeren med at vælge det tidspunkt, hvor man vil se journaldata for jf. figur 8d.

Opsummeringen og journaldataet opdateres automatisk baseret på det valgte tidspunkt, og besøgsbeskrivelsen bliver tilgængelig under journaldataet som en ny fane. Efter at have læst opsummeringen og journaldataet afgiver brugeren feedback på samme måde som for SPL-use casen. Når feedbacken er afgivet, vælger brugeren det næste besøgstidspunkt og gentager processen. Evalueringen af journalen er først færdig, når brugeren har givet feedback på opsummeringerne for alle besøgstidspunkter.

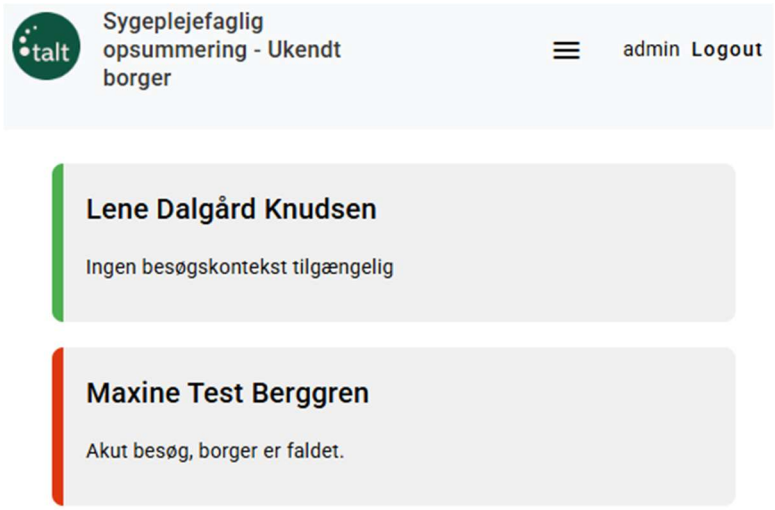
Designovervejelser

Da klienten kun skal præsentere data, så er den designet ud fra det princip, at journalen skal være overskuelig. Flowet i brugergrænsefladen følger evalueringsprocessen: øverst vises kontekst og opsummering, som brugeren skal forholde sig til først, herefter journaldataet, og nederst feedbackmodulet, der afslutter evalueringen. Klienten er lavet responsiv i forhold til skærmstørrelse, da det blev klart, at flere af kommunerne vil afprøve klienten på mobilstørrelser. Derfor følger designet mobile first-princippet.

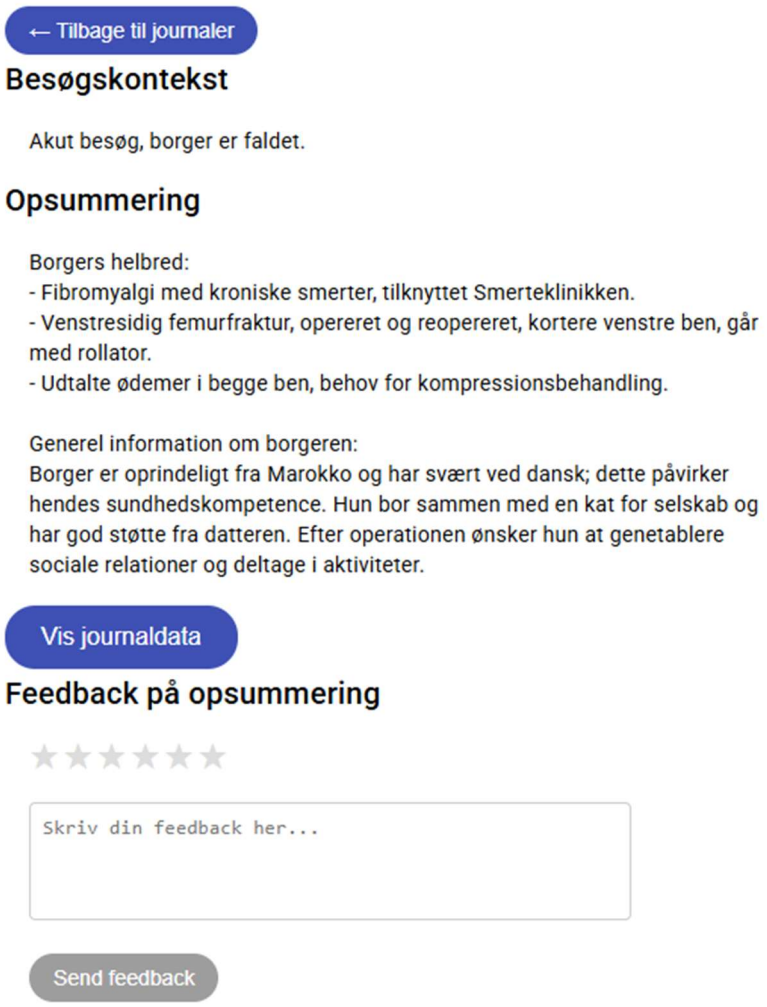
Der var bl.a. et ønske om, at journaldataet skulle være skjult på forhånd, således brugeren kun skulle forholde sig til konteksten og opsummeringen i første øjekast. Dernæst skal brugeren aktivt klikke på "vis journaldata" for at få vist datagrundlaget for opsummeringen. Dataet er delt op i sektioner i form af faner, hvor hver fane indeholder relevante borgerdata.

Opsummeringen er opstillet på punktform, hvor der er tre overordnede emner: "borgers helbred", "generel information om borgeren" og "ændringer seneste 5 dage". Dette er for at gøre det hurtigere at få et overblik af, hvad der kan være relevant.

Figur 10a: Oversigt over journaler



Figur 11b: Journalsiden for SPL-Use casen.



Bemærk: Det er ikke muligt at sende feedback, da feltet ikke er udfyldt.

Figur 12c: Journaldata sektionen hvor de forskellige sygeplejetilstande ses.

Figur 13d: Journalsiden for SSH/SSA-Use casen

Journaldata for morgenbesøgstidspunktet vises. Feedbackfelterne er udfyldt, og brugeren kan trykke "Send feedback"

Dataprocessering

Journaler blev indskrevet af fagligt spor som rækker i et Excel-ark, hvor kolonner angav journalfelter i journalen. Excel-arket blev indlæst i Python, og relevant data blev udtrukket og grupperet for at muliggøre korrekt indsættelse i sprogmodellens prompt. Ligeledes blev de fagligt-genererede opsummeringer indlæst til at tune den initiale prompt. Journalerne indeholdt information om

sygeplejetilstandene, generelle oplysninger, vigtige observationer de seneste fem dage, kliniske observationer de seneste fem dage og besøgsplaner (for SSH/SSA-løsningen).

Promptstruktur – opsummering af journaler

Promptstrukturen blev initialt lavet med henblik på at kunne genskabe de fagligt-genererede opsummeringer, og er efterfølgende blevet iterativt forbedret efter hver afprøvning ud fra en analyse af feedback fra det faglige personale.

Prompten for både spl- og ssa/ssh-opsummeringerne følger samme opbygning med følgende opskrift:

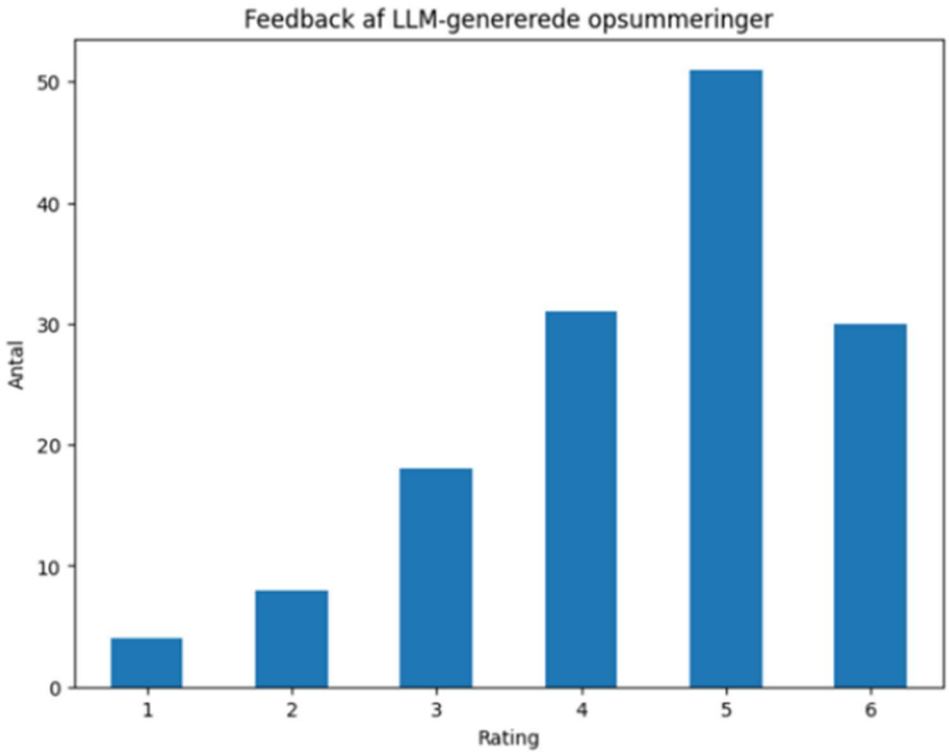
- Rolle: Beskrivelse af modellens persona og formål.
- Indholdsbeskrivelse: Beskrivelse af hvad opsummeringen skal indeholde.
- Guide: Trin for trin guide modellen skal følge.
- Regler: For hvert afsnit er der angivet specifikke regler og opmærksomhedspunkter.
- Eksempel på opsummering: Et eksempel på hvordan en opsummering kan se ud.
- Journaldata: Journaldata for den pågældende journal, der skal opsummeres.

Rolle, indholdsbeskrivelse, guide og eksempel på opsummering, er alle tilføjet for at få den rigtige form og det rigtige indhold i opsummeringen, mens reglerne er lavet for at tilpasse opsummeringerne til at ligne de fagligt genererede opsummeringer.

3.3 Evaluering

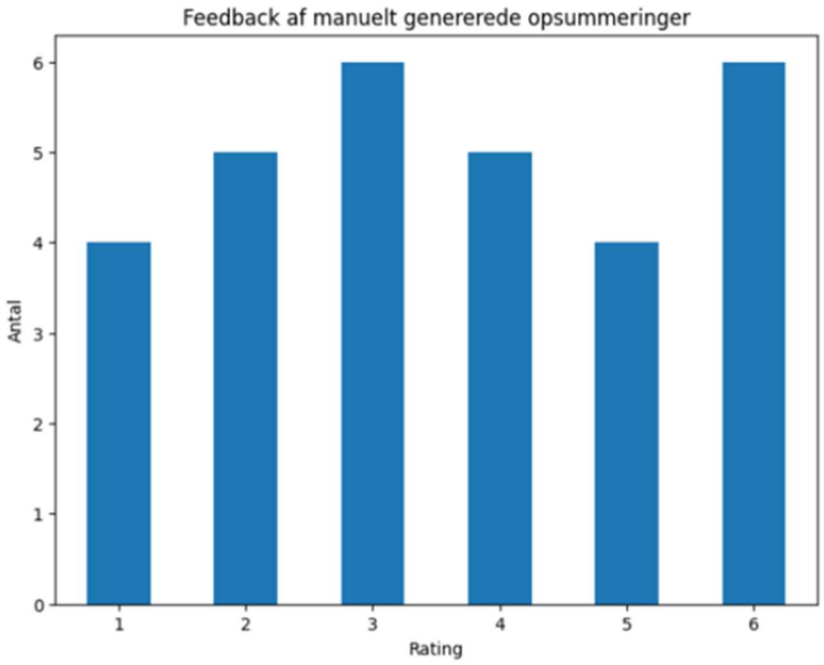
Evaluering af UC2 løsningen er udført gennem indhentning af faglige vurderinger ifm. afprøvning. Der blev afholdt en ekstern afprøvning i projekts elleve afprøvningskommuner for de sygplejefaglige opsummeringer. De gav en gennemsnitlig rating på 4,5 på en skala fra 1-6. Fordelingen af evalueringerne kan ses i figur 9a nedenfor. Seks fagligt genererede opsummeringer var også sendt med til afprøvningen, for at kunne sammenligne kvaliteten af LLM-genererede opsummeringer med fagligt manuelt-genererede opsummeringer. De fagligt genererede opsummeringer fik en gennemsnitlig evaluering på 3,6, og deres fordeling er ligeledes plottet nedenfor i figur 9b.

Figur 14a: Feedback af LLM-generede opsummering for SPL-Use casen



Feedback er givet i form af en rating fra 1-6. Feedback er indsamlet i forbindelse med afprøvning.

Figur 15b: Feedback af manuelt generede opsummering for SPL-Use casen



De manuelle opsummeringer er genereret af det sygeplejefaglige personale, til udvikling af løsningen. Feedback er givet i form af en rating fra 1-6. Feedback er indsamlet i forbindelse med afprøvning.

3.4 Videreudvikling

Videreudviklingen af denne løsning, skal bygge ovenpå det potentiale, der er blevet vist til afprøvningerne i projektet. Det anbefales at fokusere på, at vise systemet også kan yde en tilfredsstillende og værdiskabende generel opsummering med ægte journaldata og med al relevant journaldata udtrukket direkte fra den elektroniske omsorgsjournal (EOJ). Første skridt er derfor at undersøge, hvilken data er relevant for de generelle journalopsummeringer og hvorvidt det er muligt at lave en udtrækning og repræsentation af data. Når dette er på plads, kan det undersøges med samme tekniske flow som illustreret i figur 7 og med tilsvarende prompts, om løsningen kan generere værdifulde opsummeringer.

Hvis de generelle opsummeringer viser gode resultater, vil det være muligt at bygge ovenpå med besøgsspecifikke opsummeringer. Disse er rettet mod sygeplejerskerne, der typisk foretager besøg med en specifik besøgskontekst. Det vil kræve en større mængde af dokumentation fra faglig side, da sådan løsning vil kræve grundige beskrivelser af besøgskonteksterne, for at øge sprogmodellens forståelse af opgaven ved generering af opsummeringer. Ligeledes, vil det kræve fagligt genererede opsummeringer til hver af besøgskonteksterne, der kan bruges under tuningen af prompten. Der kan med fordel startes med nogle få besøgskontekster, og når disse viser lovende resultater, kan flere kobles på for at gøre processen mere iterativ.

3.5 Idriftsættelse

Dataflow og integrationer

Systemet vil enten kunne blive indbygget i EOJ'en som en ekstra feature eller i et tredjepartssystem, der trækker data fra EOJ'en og præsenterer det i en selvstændig applikation. Her vil førstnævnte være

det anbefalede setup, da løsningen i udgangspunktet er en mindre feature der i udgangspunktet er et tekstfelt med opsummeringen, og derfor burde være nem at integrere i systemet. Desuden, vil løsningen blive mere brugervenlig for personalet, hvis de ikke skal skifte frem og tilbage mellem systemer og hvis de har borgerjournalen samme sted, så de altid kan dykke ned i den, hvis ikke opsummeringen er fyldestgørende.

Uanset hvilket af ovenstående setup, der bliver valgt, skal der laves to integrationer som systemet skal bruge:

1. EOJ'en skal udstille et API, der kan trække al relevant data ud af en borgers journal til det pågældende besøg.
2. Integration til en LLM, der kan generere opsummeringen. Dette kan enten være en lokal model eller en model placeret hos en cloud provider.

Sikkerhed

Hvis systemet bliver indbygget i EOJ'en med en lokal sprogmodel, er sikkerheden sikret gennem EOJ-systemets egen datasikkerhed, som man må formode lever op til kravene omkring håndtering af følsomme personoplysninger. Bliver det derimod et eksternt system, vil der skulle udføres et større arbejde for at sikre at leverandøren af systemet lever op til gældende regler for håndteringen af sundhedsdata. Det samme vil gælde, hvis der vælges en sprogmodel placeret i skyen.

Kvalitetssikring

For at sikre tilstrækkelig kvalitet under udviklingen af systemet, er det vigtigt at få defineret både et trænings- og test-datasæt, performancemetrikker og grænseværdier for, hvor god performance skal være for, at systemet kan sættes i drift.

Et trænings- og test-datasæt skal indeholde et bredt og fyldestgørende udsnit af borgertyper med varierende kompleksitet og størrelser på journalerne for de generelle brede opsummeringer. For besøgsspecifikke opsummeringer gælder det, at alle besøgskonteksterne skal være dækket i tilstrækkelig grad i datasættene. Man kan her vælge at vurdere systemet samlet på tværs af alle besøgskontekster eller på besøgskontekstniveau, hvor det på besøgskontekstniveau kan medføre, at systemet kun bliver godkendt til at operere på udvalgte besøgskontekster, hvis nogle er for komplekse til at systemet kan opnå en god performance.

Evalueringen kan foregå ad grundlæggende to veje: en vej med manuelle faglige vurderinger og en anden med 'LLM-as-a-judge' evalueringer.
De faglige vurderinger af et systems performance vil være en langsom proces og trække på faglige ressourcer, men er formentlig den mest sikre og optimale evaluering, da personalet både kan levere en god kvantitativ evaluering, men også indgå i dialog og kvalitativt vurdere, hvad systemet mangler eller gør godt.
Med LLM-evalueringer vil den initiale opsætning være tidskrævende, men vil efterfølgende kunne levere evalueringer hurtigt og herfra kan flere iterationer af systemet udvikles med 'LLM-judgen' som ledesnor.

Da begge metoder kommer med sine egne fordele, anbefales det at kombinere disse to metoder. Dette kunne foregå ved at der først manuelt bliver lavet et trænings- og testsæt bestående af ægte journaldata og dertilhørende fagligt genererede opsummeringer. Træningssættet kan her bruges til at optimere prompten, der genererer opsummeringerne og herefter kan der blive lavet en LLM-judge til at vurdere kvaliteten af opsummeringerne ved at sammenligne fagligt-genererede opsummeringer med LLM-genererede. Dette kunne resultere i en score for hver opsummering, hvortil systemet bliver godkendt, hvis scoren for testsættet er over en bestemt grænseværdi.

I praksis vil flere specifikke opgaver implementeres i LLM-jugden, f.eks. om der optræder hallucinationer, om alle oplysninger i de faglige opsummeringer også indgår i de genererede, mængden af irrelevante oplysninger, hvorvidt formen og sproget er god nok, m.v. Her vil de alle kunne give en score, der indgår i den samlede vurdering af, hvornår systemet er godt nok.

Når der er kørt nok iterationer til at systemet godkendes af LLM-judges, kan systemet sendes til vurdering af fagligt personale. Vurderer de faglige, at det ikke er godt nok, skal deres rettelser tages med tilbage til at optimere både LLM-judgen og selve opsummeringsgenereringen. Dette træningsloop kan køre indtil, at der er overensstemmelse i vurderingerne mellem LLM-judgen og fagligt personale og systemet godkendes begge steder. Det er væsentligt for denne proces at den faglige validering foregår så tæt på ægte borgerbesøg som muligt.

Hvis der ønskes flere sikkerhedsforanstaltninger i forbindelse med genereringen af opsummeringer, kan der også i genereringsfasen tilføjes specialiserede LLM-lag, der tjekker for fx hallucinationer, form, sprog og generel kvalitet og som hver især retter opsummeringen til.

Ønsker man en dag at opdatere systemet – f.eks. hvis EOJ'en ændrer sig (får nye felter, arbejdsgange etc.) - skal trænings- og testsæt opdateres så de afspejler virkeligheden fuldt ud og dækker variationen, hvorefter opsummerings-genereringen og LLM-judgen kan genkøres og starte et nyt feedbackloop som beskrevet ovenfor.

Tidslinje

For at sætte systemet i drift vil der være nogle nødvendige undersøgelser og afklaringer der skal foretages undervejs. Nedenfor er et bud på en mulig tidslinje:

- Genkøre systemet med nye afprøvninger på ægte data og efterfølgende udvide løsningens afgrænsning til at generere besøgsspecifikke opsummeringer for løsningen til sygeplejerskerne.

For at undersøge om systemet giver mening at udvikle, skal det først undersøges, om løsningen kan give værdi på ægte data. Det skal ligeledes undersøges, om det er muligt at udtrække al relevant journaldata fra borgere gennem EOJ'en. For løsningen rettet mod sygeplejerskerne gælder det, at når systemet har påvist at kunne levere fyldestgørende generelle opsummeringer af borgerjournaler, skal man bygge ovenpå og teste det samme for besøgsspecifikke opsummeringer (se afsnittet om videreudvikling).

- Afklaring af, hvem der skal udvikle løsningen og styr på datasikkerheden.

Systemet skal sendes i udbud og en leverandør skal vælges. Kravene til datasikkerheden skal håndteres og er afhængig af, om løsningen indbygges i EOJ'en eller bygges af ekstern tredjepart i et selvstændigt system samt i hvilket miljø sprogmodellen køres.

- Afklaring af metrikker og grænseværdier for at systemet kan sættes i drift.

Det skal besluttes, hvilke metrikker og grænseværdier man vil bruge for at godkende systemet. Her kan der både bruges kontinuerlige numeriske scoringer som fx. 1-6 stjerner eller mere binære scoringer som f.eks. om opsummeringen er god nok, optræder der hallucinationer, mangler der oplysninger osv. Ved numeriske scoringer er det vigtigt klart at definere, hvad hvert trin indeholder for at få ensartede vurderinger af fagligt personale, og det ligeledes bliver nemmere for en LLM-judge at forstå de forskellige trin.

- Udarbejdelse af trænings- og testsæt.

Der skal udvælges et trænings- og testsæt med passende størrelse og variation. Dataene skal trækkes ud af EOJ'en og klargøres til LLM'en, mens fagligt personale skal generere opsummeringer, der kan tunes og evalueres op imod.

- Udvikling af sprogmodel til generering af opsummering samt LLM-judges.

En sprogmodel og eventuelle ekstra sikkerhedslag skal udvikles og tunes på træningsdataene. Den skal evalueres på testdataene, hvor LLM-judgen skal udvikles til at evaluere og score opsummeringerne.

- Systemet godkendes af både LLM-judgen og fagligt personale.

Når systemet scorer højere end den definerede grænseværdi på testdataene og dermed er blevet godkendt af LLM-judgen, skal fagligt personale køre samme vurdering af testdataene; er de enige med LLM-judgens vurderinger og scorer over grænseværdierne kan systemet sættes i drift, og ellers skal man tilbage og opdatere LLM-judgen og køre en ny tuning af opsummerings-modellen.

- Systemet sættes i drift.

Systemet kan herefter sættes i drift. Her vil det være fordelagtigt løbende at kvalitetssikre opsummeringerne. Både ved løbende at få kvalitativt feedback af personalet, som bruger systemet, men også lave stikprøve-evalueringer af opsummeringerne med fagligt personale, for at sikre systemet bliver ved med at holde en god performance over tid. Hvis kvaliteten viser sig at falde over tid eller hvis noget ændrer sig i dokumentationspraksissen, der kan have betydning for opsummeringerne, bør udviklingen af systemet genkøres med et nyt opdateret datasæt, hvortil systemet igen skal godkendes af LLM-judgen og fagligt personale for at kunne komme i drift.

4. Indtaling af besøgsbeskrivelser (UC3)

4.1 Formål og beskrivelse

Den primære problemstilling der bliver undersøgt i Use casen "Indtaling af besøgsbeskrivelser" er udfordringen med at skrive retvisende og udførlige besøgsbeskrivelser. Besøgsbeskrivelser bliver oftest skrevet efter et besøg hos en borger, muligvis flere timer eller endda dage efter, hvorfor det derfor kan være svært at huske alle detaljer omkring besøget og hvordan det skal udføres. Det er en tidskrævende proces der kan være behæftet med fejl grundet manglende tid til at skrive en udførlig besøgsbeskrivelse. I denne Use case forsøges det derfor at afklare:

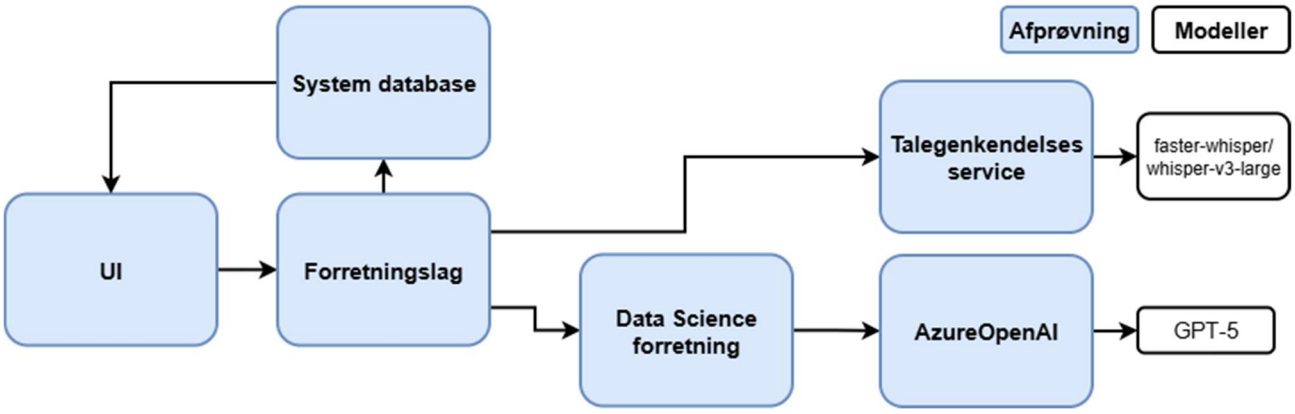
- *Hovedmål: Er det muligt for en AI-løsning at skabe en besøgsbeskrivelse af passende kvalitet, ved at modtage en dikteret besøgsplan som lyd fra en SSH/SSA?*
    - *Delmål: Er det muligt for en AI-løsning at opdatere/rette en eksisterende besøgsplan, baseret på en diktering af de ønskede ændringer fra en SSH/SSA?*

Ved at automatisere genereringen af besøgsbeskrivelsen, håber vi på at opnå en mere retvisende besøgsbeskrivelse, da den kan genereres kort tid efter et eventuelt besøg, samt at øge kvaliteten og konsistensen. Det er også muligt at der kan opnås gevinst i form af effektivisering, da tiden det tager at skrive/rette i en besøgsbeskrivelse kan reduceres. Dette var dog ikke det primære formål med at undersøge denne Use case, og derfor er dette heller ikke reflekteret i målene for afprøvningen af den udviklede løsning.

4.2 Teknisk løsning

Løsningens til understøttelse af Use casen bygger på en lignende afprøvningsklient som for Use case 1. Løsningens genbruger mange af de samme komponenter, der undervejs i projektet blev udvidet til at understøtte afprøvning af flere Use cases ad gangen. Se figur 10 for en oversigt over komponenterne i UC3 løsningen, bemærk det store overlap med løsningen for UC1 (figur 1).

Figur 16: Indtaling af besøgsbeskrivelse - Komponent oversigt



Løsningen anvender vores talegenkendelsesservice (se afsnit 3.2 for uddybende afsnit) til at transskribere indtalingen. Dernæst genereres en passende besøgsbeskrivelse på baggrund af transskriptionen med en stor sprogmodel (GPT-5[10] ved afprøvning) via Data Science forretningslaget.

[10] GPT-5 Model | OpenAI API

Formatet for denne besøgsbeskrivelse er udarbejdet i samarbejde med de tre interne afprøvningskommuner, og bliver givet til sprogmodellen i form af en systemprompt der indeholder formatet samt øvrige instrukser til at generere besøgsbeskrivelsen. Den genererede besøgsbeskrivelse bliver derefter udstillet til brugeren igennem afprøvningsklienten, hvor besøgsbeskrivelsen kan rettes til og der kan gives feedback på outputtet. Ligesom de øvrige Use cases i TALT projektet, er den primære form for undersøgelse af hypoteserne afviklet igennem afprøvninger med potentielle brugere fra afprøvningskommunerne. Anderledes fra de øvrige Use cases har afprøverne dog udelukkende en SSH/SSA-baggrund.

Som følge af delmålet omkring indtaling af opdateringer blev der udviklet en mindre udvidelse til afprøvningsklienten, der udnyttede de samme komponenter som tidligere til at skabe besøgsbeskrivelser ved hjælp af AI. Den eneste forskel var systemprompten der blev brugt til at indføre opdateringerne i den allerede eksisterende besøgsplan.

Afprøvningsklient

Afprøvningsklienten er struktureret omkring at lave besøgsplaner og evaluere dem: oprettelse af nye besøgsplaner, visning og redigering af genererede planer samt opdatering af eksisterende planer med nye ændringer. Funktionaliteten for use-casen er følgende:

- Opret besøgsbeskrivelse (hovedmål):
  - Live optagelse
  - Indlæsning af lydfil
- Opdatering af eksisterende besøgsbeskrivelse (delmål):
  - Live optagelse
  - Indlæsning af lydfil
- Mulighed for manuel retning af tekst
- Feedback-modul

Klientbeskrivelse

Afprøvningsprocessen starter fra en oversigtsside, der viser brugerens besøgsbeskrivelser. Hver besøgsbeskrivelse i listen præsenteres med navn, besøgstidspunkt og en farvekodning, der indikerer status: rød for nyoprettede besøgsbeskrivelser, gul for besøgsbeskrivelser klar til opdatering, grøn for færdige planer og blå, når systemet er ved at processere en besøgsbeskrivelse jf. figur 11a. I toppen af siden har brugeren mulighed for at oprette en ny besøgsbeskrivelse via knappen "opret besøgsplan/døgn-rytmeplan" jf. figur 11b.

Når en bruger opretter en ny besøgsplan, skal to felter udfyldes: et navn på besøgsplanen samt valg af besøgstidspunkt (morgen, middag, aften eller nat). Efter udfyldning skal brugeren vælge enten at indtale besøgsbeskrivelsen live ved at optage direkte i browseren eller at indlæse en eksisterende lydfil. Hvis brugeren laver en live indtagelse, så skal personen trykke på start og stop optagelse. Når lyden er optaget eller uploadet, vises en spinner-dialog, der indikerer, at systemet arbejder på transskriberingen og genereringen af besøgsbeskrivelsen. Når processen er færdig, navigeres brugeren automatisk til visningen af den genererede besøgsplan.

Besøgsbeskrivelsen er i et struktureret format med sektioner som personlig pleje, ernæring, hverdagsaktiviteter og medicin. Formatet er udarbejdet i samarbejde med de tre afprøvningskommuner. Brugeren har mulighed for at se den oprindelige transskribering af deres optagelse ved at klikke på "Vis transskribering", hvilket gør det muligt at verificere, om eventuelle fejl

skyldes fejl i talegenkendelsen eller i LLM'ens fortolkning. Brugeren har mulighed for at rette i besøgsplanen, før en evt. opdatering laves.

Under hver genereret besøgsplan findes feedbackmodulet, hvor brugeren kan give en stjernebedømmelse (1-4 stjerner) samt skrive en tekstkommentar jf. figur 11c. Feedbacken gemmes sammen med besøgsplanen og muliggør senere analyse af besøgsbeskrivelser. Når feedbacken er sendt, navigeres brugeren til siden, hvor opdateringen kan indtales, jf. figur 11d. Opdateringsprocessen følger samme flow som ved oprettelse: brugeren dikterer eller uploader en lydfil med de ønskede ændringer, hvorefter systemet transskriberer og genererer en opdateret version.

Når brugeren har lavet en opdatering til den eksisterende besøgsplan, vises den i venstre side af skærmen som reference, mens den opdaterede version bliver vist i højre side jf. figur 11e. Brugeren har igen mulighed for at rette besøgsbeskrivelsen, og der skal gives feedback som tidligere. Når der er givet feedback, er evalueringen slut for denne besøgsbeskrivelse, og brugeren bliver sendt ud til listen af besøgsbeskrivelser. Flowet starter nu forfra, hvor brugeren skal lave en helt ny besøgsplan.

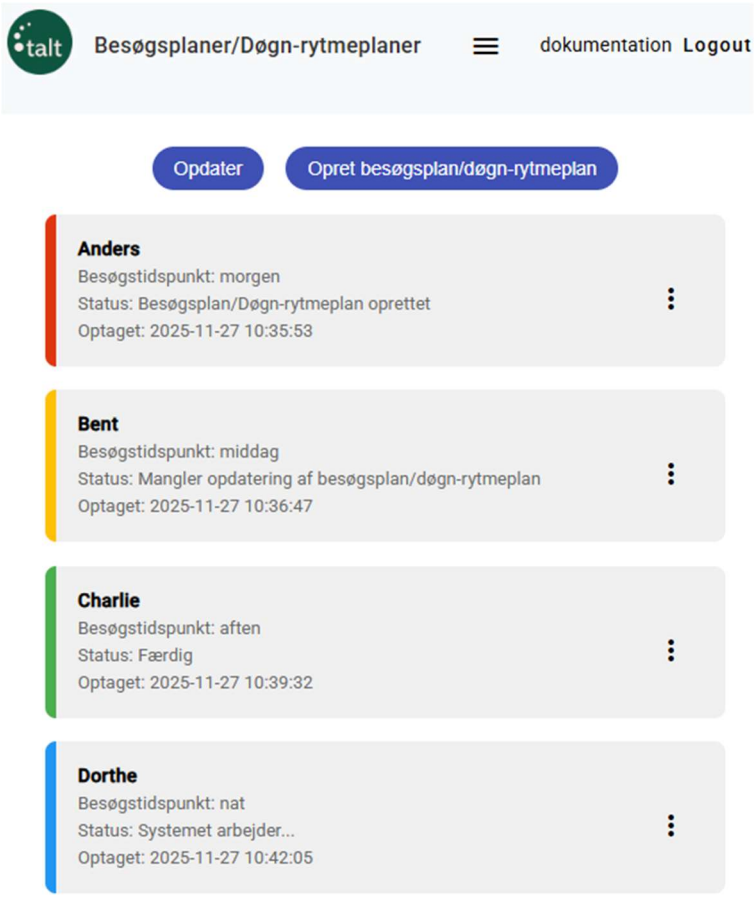
Designovervejelser

Den oprindelige vision var, at det skulle være muligt at indtale en besøgsbeskrivelse, som systemet skulle omskrive til et struktureret format. Brugergrænsefladen skulle supportere, at man kunne indtale eller indlæse en lydfil, således denne kunne sendes til systemet, som ville generere en besøgsbeskrivelse. Dernæst skulle fagpersonalet give feedback på besøgsbeskrivelsen. Planen var således, at systemet skulle designes omkring selve besøgsbeskrivelsen.

Ønsket fra det faglige personale var dog, at brugergrænsefladen skulle kunne håndtere, at man skulle kunne rette i besøgsbeskrivelsen, og at der var mulighed for at opdatere en besøgsbeskrivelse vha. en indtalelse. Det ledte til, at der på den tekniske side blev introduceret konceptet samling. En samling består af flere besøgsbeskrivelser, hvor hver har en bestemt type. Denne type beskriver, hvilken slags besøgsbeskrivelse det er, og der er følgende typer: original, rettet, opdateret, rettet opdatering. Samlingen indeholder vigtig information såsom, hvem der har lavet den, hvornår, besøgstidspunktet og om systemet er i gang med at processere en besøgsbeskrivelse. Samlingen gør det desuden nemmere at lave analyser på, hvor lang tid hele processen har taget, og eftersom der ligger timestamps for hver oprettet besøgsbeskrivelse, så er det muligt at få en finere granularitet.

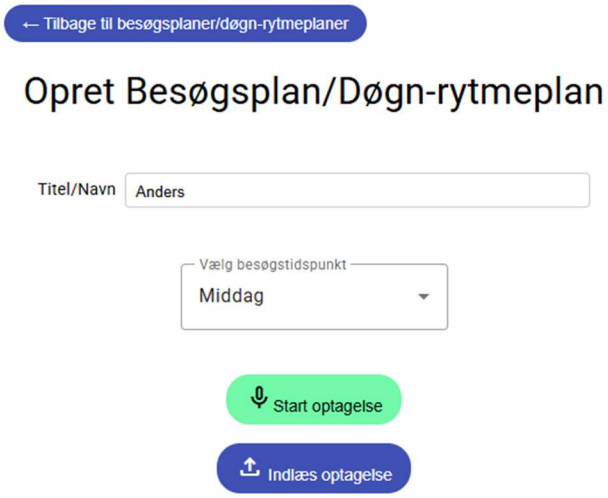
Systemet processerer en besøgsbeskrivelse asynkront, så det betyder, at brugeren selv skal opdatere listen af besøgsbeskrivelser, ligesom i Use case 1. Hvis brugeren stadigvæk er på siden, hvor besøgsplanen bliver genereret eller opdateret, så polles backenden hvert 5. sekund for at tjekke, om processen er færdig. Dette er for at håndtere de scenarier, hvor det tager længere tid at processere en besøgsbeskrivelse. Hvis processen fejler undervejs, så bliver alt rullet tilbage i databasen. Ved en nyoprettet besøgsbeskrivelse bliver samlingen markeret som fejlet, og den vil ikke kunne ses i brugergrænsefladen. Ved en opdatering, der fejler, så rulles der tilbage til det stadie, hvor brugeren kan lave en ny opdatering, så der ikke skal laves en ny original besøgsbeskrivelse.

Figur 17a: Besøgsplanoversigten for en bruger

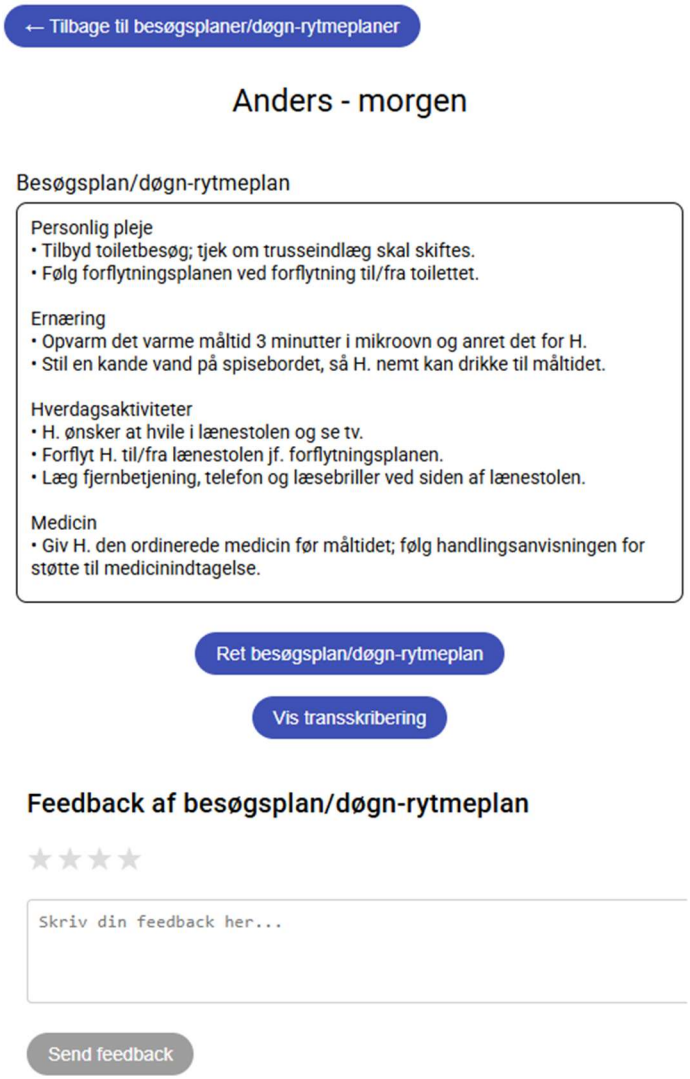


Farverne indikerer hvor i evalueringsprocessen brugeren er samtidigt med statusteksten.

Figur 18b: Oprettelse af ny Besøgsplan

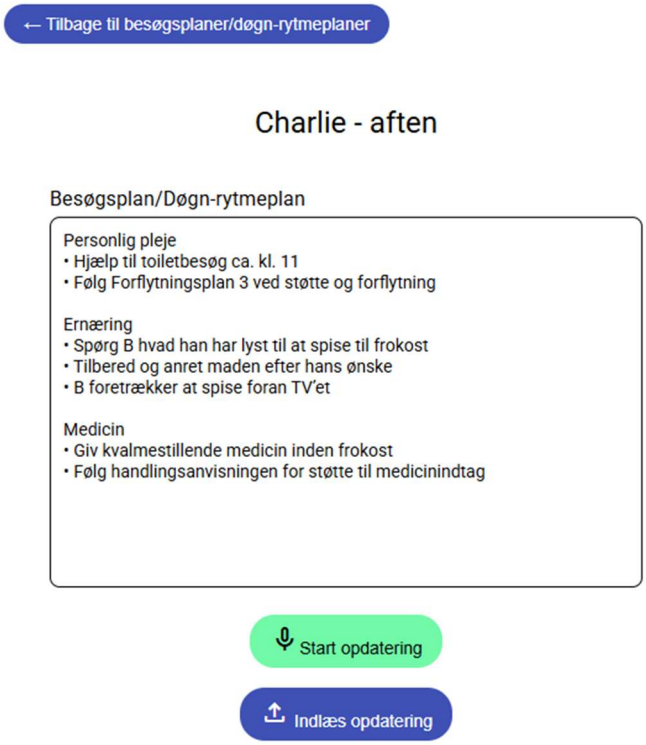


Figur 19c: Visning af den generede besøgsbeskrivelse.

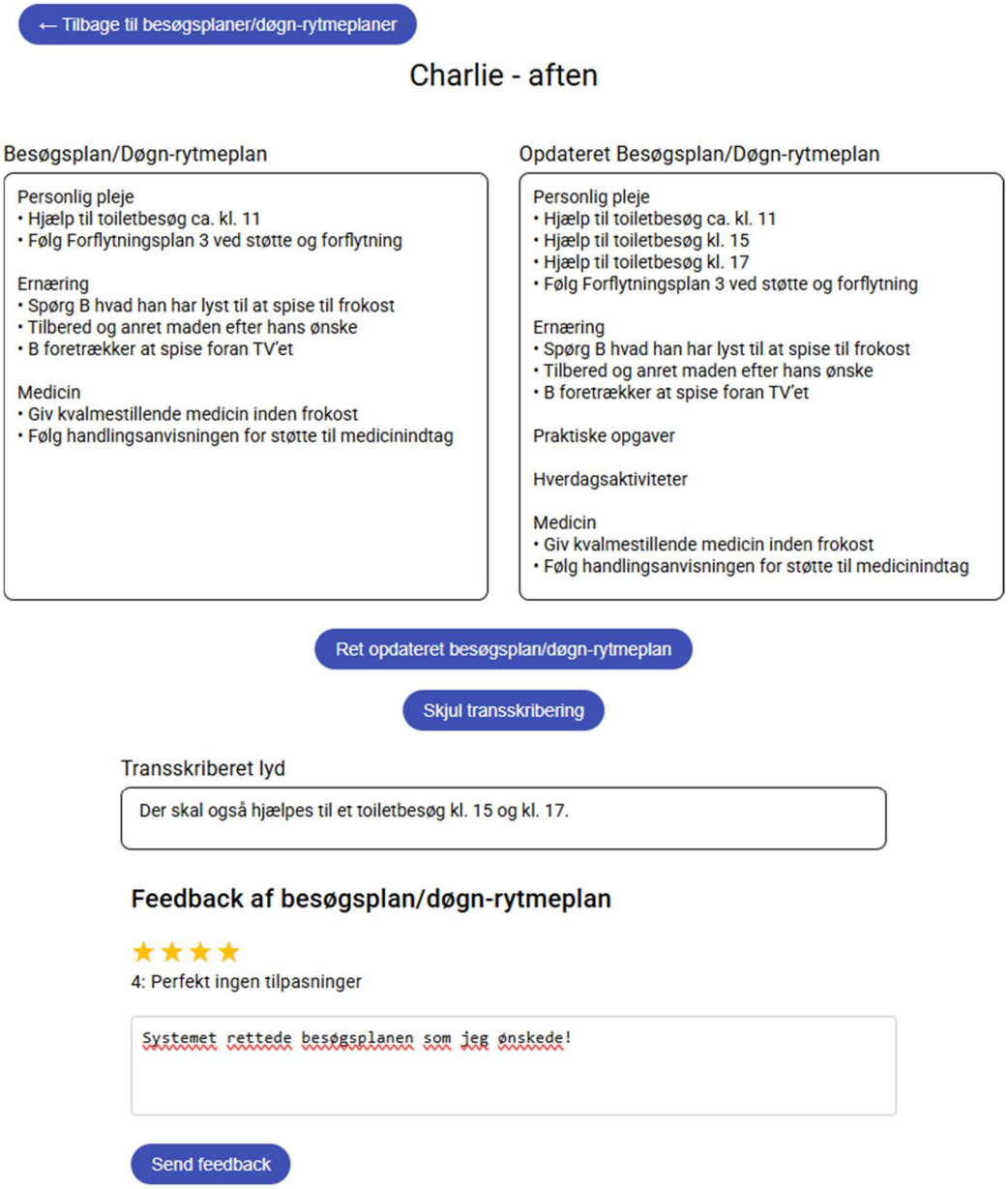


Brugeren kan se transskribering af samtalen. Dette giver kontekst til hvordan LLM'en har generet besøgsbeskrivelsen.

Figur 20d: Indtal/indlæs en opdatering til besøgsbeskrivelsen



Figur 21e: Opdateringssiden



Brugeren har optaget en opdatering og systemet har generet en opdateret besøgsbeskrivelse og der er givet feedback på den.

Opsummering og opdatering - Promptstruktur

Opsummeringskomponentens promptstruktur til generering af besøgsplaner er udformet til at håndtere en transskription af en monolog indtalt af social- og sundhedsassistent eller hjælper (SSA/SSH). Personalet indtaler i daglig tale og uden større hensyn til struktur den information der skal fremgå af besøgsbeskrivelsen.

Prompten er struktureret til at arbejde med en rå, relativt ustruktureret transskription, og har til formål at:

- Filtrere støj fra (smalltalk, irrelevante udsagn).
- Ekstrahere de fakta, der er relevante for besøgsplanen.
- Kondenserer disse fakta til korte, handlingsorienterede punktformer.
- Sikre at sproget er: dansk, nutid, i imperativ (handlingsverber som "hjælp", "husk", "stil …", osv.).

Komponenten har to primære funktioner:

- **Initial generering (system_prompt):** Givet en transskription af en indtalt besøgsbeskrivelse danner komponenten en ny besøgsplan med en fast struktur "Personlig pleje", "Ernæring", "Praktiske opgaver", "Hverdagsaktiviteter", "Medicin". Outputtet er kort, skrevet i et let forståeligt hverdagssprog (med sygeplejefaglige termer hvor nødvendigt) og fokuserer på, hvilken støtte der gives, og hvordan den leveres, med udgangspunkt i borgerens ressourcer og ønsker. Kun overskrifter med relevant indhold medtages. Se bilag A for promptstrukturen.
- **Opdatering (update_prompt):** Givet en eksisterende besøgsplan og en ny transskription opdaterer komponenten udvalgte dele af planen. Opgaver som ikke længere skal udføres, bliver fjernet, Nye eller ændrede opgaver, der nævnes i transskriptionen, tilføjes eller rettes, og alle øvrige afsnit forbliver uændrede. Overskrifter uden resterende relevant indhold fjernes. Se bilag B for promptstrukturen.

Begge prompts deler en fælles struktur:

- **Rolle:** instruerer LLM'en i at være en erfaren social- og sundhedsassistent (SSH/SSA) i hjemmeplejen.
- **Opgave:** beskriver opgaven der skal udføres, om der skal genereres en ny plan, eller om en eksisterende skal opdateres.
- **Regler:** definerer domænespecifikke krav til indhold (rehabiliterende, forebyggende og kompenserende tiltag), hjælpeniveau, kronologisk rækkefølge, tiltag på bestemte ugedage, brug af korrekte danske fagtermer og streng afhængighed af transskriptionen (ingen opdigtet information).
- **Struktur:** oplister de tilladte overskrifter og deres tilsigtede indhold, hvilket sikrer et ensartet skema på tværs af alle planer.
- **Input:** angiver eksplicit, at planen kun må baseres på den givne transskription (og den eksisterende plan ved opdateringer), og at hallucinationer og gentagelser i transskriptionen skal ignoreres.

4.3 Evaluering og afprøvning

Til at evaluere løsningen er der udført først en intern- og dernæst en ekstern afprøvning af systemet hvor brugerne har vurderet løsningen og kvaliteten af funktionaliteten.

Intern afprøvning

Den interne afprøvning af systemet blev foretaget af de tre kommuner med repræsentanter i fagligt spor: København, Aalborg og Aarhus. Da dette blev gjort tidligt i forløbet for udviklingen af løsningen, blev denne afprøvning foretaget asynkront; lydfiler med indtalinger af besøgsplaner blev leveret til teknisk spor et par dage før afprøvning og disse blev behandlet og derefter udstillet igennem afprøvningsklienten så de kunne evalueres af fagligt spor.

Evalueringerne fra den interne afprøvning bliver ikke gennemgået i detaljer i dette afsnit, da mængden af feedback var væsentligt mindre end ved den eksterne afprøvning. Desuden var der primært et altoverskyggende forbedringsområde: Den manglende struktur på de genererede besøgsplaner. Årsagen til dette skal findes i at LLM'en ikke var blevet instrueret til at følge en stringent besøgsplan, da der var forskellige ønsker til hvordan strukturen skulle være for en besøgsplan. Resultaterne fra afprøvningen ledte til at der fra fagligt spors side blev udarbejdet et forslag til en struktur, som derefter blev indført i prompten til LLM'en. Derefter blev der kørt flere forbedringsiterationer hvor fagligt spor kom med input til de genererede besøgsplaner, som kunne indføres i prompten til LLM'en. I sidste ende blev der derigennem opnået en tilstrækkelig kvalitet af de genererede besøgsplaner til at det blev vurderet at det gav mening at undersøge funktionaliteten igennem en ekstern afprøvning.

Sideløbende med forbedringerne til generering af besøgsplaner, blev der undersøgt hvorvidt LLM'en var i stand til at tage en eksisterende besøgsplan og foretage en opdatering af denne, ud fra en transskription der beskrev hvad der skulle ændres i besøgsplanen. Denne funktionalitet blev ikke testet i den interne afprøvning, men fra de tidlige tests af featuren vurderede både fagligt og teknisk spor at den var tilstrækkeligt raffineret til at blive testet i en ekstern afprøvning som en forlængelse af "Indtalelse af besøgsbeskrivelser" funktionaliteten.

Ekstern afprøvning

Ekstern afprøvning blev udført fra d. 17 til 28 november 2025 af de 11 kommuner fra afprøvningssporet. I tabel 7 kan der ses en oversigt over antallet af oprettede og opdaterede besøgsplaner for de forskellige kommuner. I denne oversigt, samt senere analyser, er der foretaget en frasortering af de besøgsplaner der ved en fejl er oprettet i systemet. Der var desuden en misforståelse af hvordan systemet skulle anvendes ved afprøvning i Odense kommune, hvorfor en del af data og dertilhørende evalueringer er sorteret fra. Egedal kommune oplevede også problemer med de devices der skulle bruges til at optage besøgsplanerne, hvorfor det kun lykkedes dem at generere et meget begrænset antal besøgsplaner. I alt er der sorteret lidt over 40 oprettede/opdaterede besøgsplaner fra, hvor de fleste er sorteret fra pga. manglende lyd eller fordi optagelsen er begyndt ved en fejl. Derudover var der også nogle brugere der, i et forsøg på at gøre besøgsplanen "grøn" i frontenden, lavede opdateringer til besøgsplaner uden reelt at bede om nogle ændringer. Disse opdaterede besøgsplaner er også blevet sorteret fra.

Tabel 6: Oversigt over antallet af oprettede og opdaterede besøgsplaner

Kommune	Antal oprettede besøgsplaner	Antal rettede besøgsplaner	Antal opdaterede besøgsplaner	Antal rettede opdaterede besøgsplaner
Aalborg	25	15	23	8
Aarhus	16	7	11	5
Egedal	6	0	1	0
Haderslev	10	3	9	2
Hjørring	16	6	14	4
Horsens	22	13	9	4
København	33	16	24	13
Odense	10	3	3	3
Roskilde	15	9	6	4
Vejle	17	1	14	4
Viborg	13	3	13	3
Total	183	76	127	50

Ud fra tabellen kan det ses at ca. 2/3-dele af oprettede besøgsplaner blevet opdateret, hvilket giver et nogenlunde fornuftigt grundlag til at vurdere kvaliteten af både " Indtalelse af besøgsbeskrivelse" "Opdatering af besøgsplan" funktionaliteten. Lige omkring 60 % af både oprettede og opdaterede besøgsplaner er ikke blevet rettet til efter oprettelse; dette er undersøgt ved et simpelt check af tekststrengen for den rettede besøgsplan med den originalt oprettede besøgsplan. En mere dybdegående analyse af ændringerne til besøgsplanerne kan læses længere nede.

Man skal være påpasselig med at konkludere at LLM'en kan generere "perfekte" besøgsplaner mere end halvdelen af tiden dog, da det er muligt at de nødvendige rettelser til besøgsplanen ikke er blevet indført i et afprøvningsscenarie (på trods af at dette har været intentionen). Den relativt høje procentsats indikerer dog en vis kvalitet af de oprettede besøgsplaner, og det må kunne forventes at der vil være et antal af besøgsplanerne der kunne anvendes uden rettelser i et produktionsmiljø.

En indikator for kvaliteten af besøgsplanerne kan også ses på den gennemsnitlige rating/evaluering givet af kommunerne i tabel 8. Generelt ses det at besøgsplanerne kan bruges med få rettelser (3 stjerner). Det ser også ud som om LLM'en generelt kan finde ud af at løse opgaven med at opdatere den eksisterende besøgsplan med rimelig præcision, og den gennemsnitlige rating er væsentligt højere for opdaterede besøgsplaner end for de oprindeligt genererede. Især den høje gennemsnitlige rating for Egedal kommune skal dog tages med forbehold, da det kun er baseret på de 6 genererede besøgsplaner.

Tabel 7: Gennemsnitlig rating for afprøvning for hver kommune.

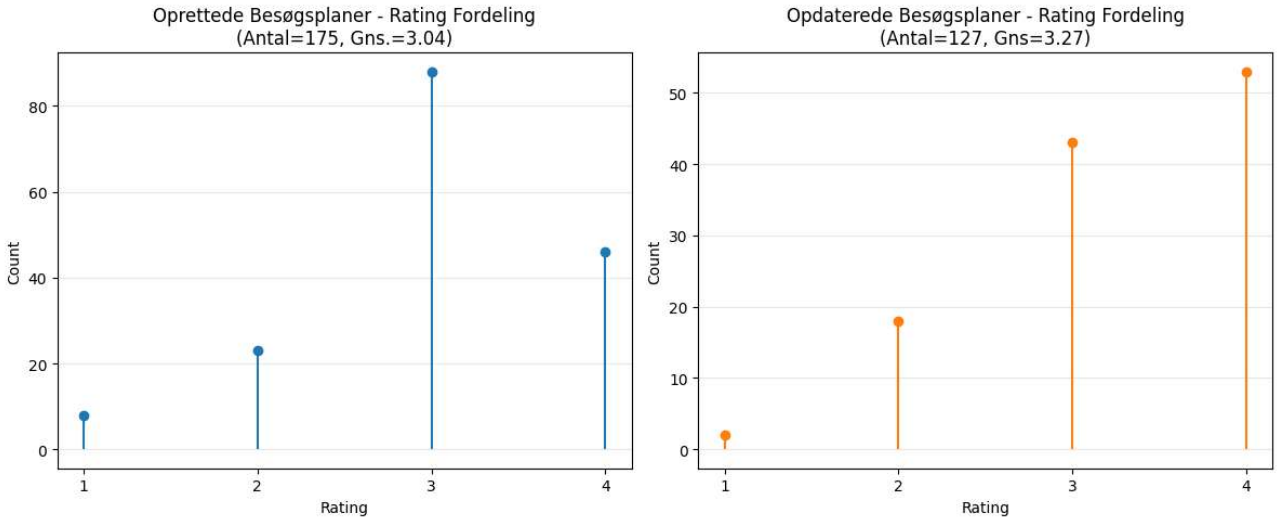
Kommune	Gennemsnitlig rating (originale besøgsplaner)	Gennemsnitlig rating (opdaterede besøgsplaner)
Aalborg	3,08	3,43
Aarhus	2,50	2,90
Egedal	4,00	3,00
Haderslev	3,10	3,11

Hjørring	3,21	2,83
Horsens	2,71	3,22
København	3,31	3,52
Odense	2,91	2,00
Roskilde	3,33	3,00
Vejle	3,06	3,50
Viborg	3,00	3,45
Total	3,04	3,27

Bemærk: Dette gennemsnit er udregnet ud fra de besøgsplaner der er blevet givet en rating.

Ser man nærmere på fordelingen af ratings i figur 12, bliver det mere tydeligt at der generelt var stor tilfredshed med de opdaterede besøgsplaner fra afprøvningssporet. Langt størstedelen af både oprettede og opdaterede besøgsplaner opfattes som værende brugbare, da disse rapporteres som enten "Kræver få rettelser" eller "Perfekt, ingen tilpasninger" (rating 3 eller 4). Ud fra graferne ses det at ca. 25 % af oprettede besøgsplaner er vurderet som rating 4, hvilket muligvis er en bedre indikator for kvalitetsniveauet end antallet af rettede besøgsplaner. Tilsvarende har ca. 40 % af opdaterede besøgsplaner opnået rating 4.

Figur 22: Fordeling af ratings for oprettede og opdaterede besøgsplaner

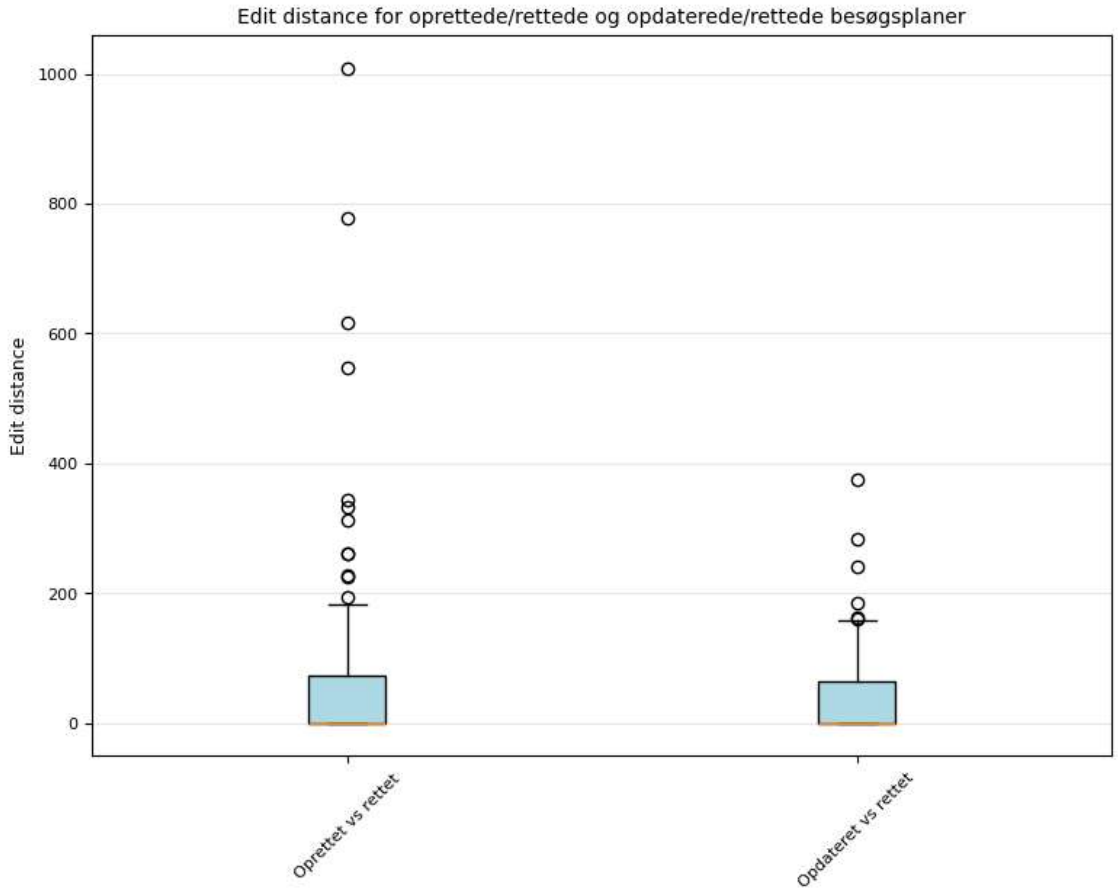


Levenshtein edit distance

For at få en bedre idé om de fejl og mangler der kan være for AI genererede besøgsplaner, kan man sammenholde de oprettede og opdaterede besøgsplaner med de tilsvarende rettede besøgsplaner. En metrik som kan bruges til dette er "Levenshtein distance", der er et mål for antallet af tegn der skal ændres for at den oprettede besøgsplan bliver til den rettede udgave. En større distance er derfor et mål for en større ændring i besøgsplanen.

På figur 13 er Levenshtein distancen plottet for de besøgsplaner der indeholder ændringer. Det kan ses at der overordnet set bliver lavet relativt små ændringer til både oprettede og opdaterede besøgsplaner. Mere end halvdelen af både oprettede og opdaterede besøgsplaner der er blevet ændret, har ændringer på mindre end 100 tegn, og desuden er langt de fleste ændringer under 200 tegn. Som forventet ud fra de højere ratings på opdaterede besøgsplaner, er ændringerne mindre omfangsrige generelt.

Figur 23: Boxplot for Levenshtein distance for oprettede og opdaterede besøgsplaner



Bemærk: Levenhstein distancen er udregnet på oprettede og opdaterede besøgsplaner der er blevet ændret, sammenlignet med deres rettede versioner. Se tabel 15 i bilag C for relevante tal.

Klassificering af fejl

For at få mere indsigt i de typer af fejl som systemet begår når der skal skrives besøgsplaner for Use case 3, er der udført en klassificerings analyse ved at udnytte en LLM. Baseret på erfaringer fra især Use case 1 var der på forhånd nogle forventninger til de typer af fejl der kunne opstå i et system der anvender tale-til-tekst- og store sprog-modeller. Fra Use case 1 blev der især set fejl af disse typer:

- Hallucinationer
- Tale-til-tekst fejl/transskriberingsfejl
- Terminologifejl

For at få en mere finkornet analyse af fejlene, blev der tilfældigt udvalgt 30 oprettede besøgsplaner med ≤3 rating, og transskriberingerne, rettelserne og besøgsplanerne blev sendt til en GPT 5.1 instans der blev instrueret i at identificere fejltyper ud fra informationen. Ud fra disse besøgsplaner blev der foreslået følgende 6 kategorier, plus "Ingen fejl" og "Andet", for at sikre imod rækker hvor ingen af kategorierne gav mening:

- Transskriberingsfejl

- Hallucinationer
- Terminologifejl
- Udeladelse af information
- Grammatiske fejl
- Kvantificerings fejl (forkerte numeriske detaljer, timing)
- Ingen fejl
- Andet

Med de 8 kategorier som del af instruksen, blev GPT 5.1 bedt om at identificere om en eller flere af disse fejl var årsag til de rettelser som blev indført i besøgsplanen. Se tabel 9 for en oversigt over fremkomsten af de forskellige typer af fejl.

Tabel 8: Typer af fejl i oprettede besøgsplaner med rettelser fortaget (76 i alt)

FEJLTYPE	ANTAL	PROCENT (%)
HALLUCINATION	40	52,6
TRANSSKRIBERINGSFEJL	38	50
TERMINOLOGIFEJL	33	43,4
UDELADELSER	32	42,1
GRAMMATIK & STAVNING	16	21,1
KVANTIFICERINGSFEJL	15	19,7
INGEN FEJL	6	7,9
ANDET	2	2,6

Den oftest forekomne fejl er hallucination, der forekommer i lige over halvdelen af de oprettede besøgsplaner. Dernæst kommer transskriberingsfejl og terminologifejl med tilsvarende frekvens, de samme fejl som blev set begået ofte i UC1 for et lignende system. Lidt overraskende, mener LLM'en at der også relativt ofte er udeladelser af information. Dette kan være et resultat af at den LLM der genererede besøgsplanen har vurderet at dele af informationerne ikke passede ind i den struktur der var bestemt. Det kan også være et område man skal være opmærksom på ifht. kvalitetssikring, så brugeren kan blive præsenteret for noget af den udeladte information. De sjældnest forekomne fejl er grammatiske- og kvantificeringsfejl, der oftest vil være nemmere for brugeren at fange, selvom kvantificeringsfejl godt kan være graverende hvis tidspunkter eller andre tal er ukorrekte.

Hallucinationer er den mest alvorlige fejl i systemet, da den som fejltype minder mindst om de fejl som et menneske ville lave, og det desuden er den fejl der opleves oftest. De øvrige fejl er nok fejl der allerede eksisterer for besøgsplaner skrevet af mennesker; ting kan blive hørt eller nedfældet forkert, og der kan være byttet rundt på tidspunkter og tal. Disse ting er alvorlige, men hallucinationer kan i værste tilfælde være skadelige og have stor indflydelse på borgerens ve og vel. Det er derfor helt essentielt at denne type af fejl bliver fanget og udbedret så ofte som muligt.

Feedback topic clustering

For at få yderligere indsigt i de typer af fejl som systemet laver, kan man nærstudere den feedback der er givet af afprøvningssporet. Da der kan være forskel på den type af feedback der associeret med hhv. oprettede og opdaterede besøgsplaner, bliver der anvendt "topic clustering" på hvert datasæt. Til at analysere feedback er der anvendt BERTopic, i et forsøg på at finde gennemgående

temaer i feedbacken. Se tabel 10 og 11 for optælling af temaerne i hhv. oprettede og opdaterede besøgsplaner.

Tabel 9: Temaer i feedback tekster på oprettede besøgsplaner

OPRETTEDE BESØGSPLANER ANTAL FEEDBACK TEKSTER	TEMA
39	Mismatch i dokumentation
24	Unødvendige og forkerte ord
20	Mangelfuld og upræcis transskription
17	Kvalitet af tekstopsætning

Tabel 10: Temaer i feedback tekster på opdaterede besøgsplaner

OPDATEREDE BESØGSPLANER ANTAL FEEDBACK TEKSTER	TEMA
33	Uklarhed og overflødig tekst
15	AI tekstforbedring og fejl
12	Rettelser og rækkefølge

Ser man på temaerne både for oprettede og opdaterede besøgsplaner, ses det at der er flere temaer som går igen i begge datasæt. Der er en del feedback omkring at besøgsplanerne generelt er for teksttunge og mangler præcision, da sprogmodellen bruger forkerte ord, højst sandsynligt sygeplejefaglige ord den ikke kender til. Omkring mængden af te er større sprogmodeller er mere verbose end mindre, noget som også blev set i analyserne for Use case 1, og da der bruges en af de større sprogmodeller til Use case 3, var det en mulig risiko. En stor del af den manglende præcision stammer højst sandsynlig fra transskriberingen der ofte fejltolker visse ord, noget som er endnu sværere når der er tale om specifik sygeplejefaglig terminologi. Dette kan også stamme fra LLM'en der heller ikke nødvendigvis kender alt den korrekte terminologi.

For oprettede besøgsplaner er der et tema dedikeret til mangelfuld transskribering, hvilket giver mening, da indtalingerne som oftest er længere og indeholder mere komplicerede instrukser, der derfor giver anledning til flere fejl i transskriberingen.

Der er også et tema omkring strukturen på besøgsplanerne og rækkefølgen/kronologien for instrukserne. Dette er heller ikke helt uventet, da forskellige kommuner har forskellige krav til hvad der skal stå i besøgsplanen og hvordan. Den struktur der er valgt fra fagligt og teknisk spors side, kan derfor ikke bruges i alle kommuner, og i et produktionssystem vil det være nødvendigt at skræddersy strukturen til at møde den kommunes behov.

Det vigtigste gennemgående tema vedrører mismatch i dokumentationen og fejl begået af LLM'en. Som oftest drejer det sig om hallucinationer fra den store sprogmodel, der opfinder instrukser eller digter videre på de instrukser der bliver givet af brugeren, på trods af at dette ikke er instrueret. Et eksempel kan f.eks. være når borgeren skal tage medicin; i disse tilfælde hallucinerer sprogmodellen ofte at SSA/SSH'en skal tjekke at borgeren har taget sin medicin, selvom dette ikke er instrueret. Hallucinationer er helt sikkert den største udfordring ved at bruge store sprogmodeller, og når disse systemer skal sættes i produktion, vil det være nødvendigt at have flere sikkerhedsforanstaltninger der minimerer antallet og størrelsen af de hallucinationer modellen bringer ind i besøgsplanen.

Udførselstider

Udover analyser af kvaliteten af de oprettede besøgsplaner er der også set på tiden det tog at generere besøgsbeskrivelserne, fra lydfilen sendes ind til backenden af afprøvningsklienten til besøgsbeskrivelsen er udstillet til brugeren. Se tabel 12 for oversigt over tiden.

Tabel 11: Beregnings og udførselstider på "Indtaling af besøgsbeskrivelser"

	Udførselstid (s)	Længde af indtaling (s)
Gennemsnit	39.3	89.1
Standardafvigelse	34.0	65.0
Median	24.8	72.1
Minimum	7.4	9.00
Maximum	191	573

Der er en relativt stor spredning på både længden af indtalinger og udførselstiden for genereringen af besøgsbeskrivelsen. I gennemsnit ligger indtalingerne på omkring de 1½ til 2 minutter og udførselstiden ligger på omkring 40 sekunder. Udførselstiden er ikke udelukkende afhængig af længden af indtaling da den også afhænger af belastningen på systemet under de forskellige afprøvninger. Der er desuden en ikke lineær sammenhæng imellem længden af lydfilen og tiden det tager at transskribere en samtale, som også blev set i Use case 1; generelt er faster-whisper forholdsmæssigt meget hurtigere til at transskribere lange lydfiler end korte. Da der bliver brugt en større sprogmodel til genereringen af besøgsplaner i Use case 3 (GPT o3 i stedet for Llama 3.3 70B) tager genereringen af besøgsbeskrivelsen længere tid end de enkeltvise opsummeringer i Use case 1.

Hvis man sammenholder udførselstid med længden af indtalte besøgsbeskrivelser i figur 14, ses der ikke umiddelbart en tydelig sammenhæng. For eksempel har både Viborg og Horsens kommune relativt lange indtalinger, men dette giver ikke anledning til signifikant længere udførselstider end de øvrige kommuner. Omvendt har Aarhus og Vejle relativt korte indtalinger men en længere udførselstid. Svaret skal nok findes i belastningen af systemet på afprøvningsdagene. Hvis man ser på tabel 13 er det de kommuner der har afprøvninger oveni andre kommuner der har lange udførselstider, hvilket skyldes kødannelser da systemet ikke er skaleret tilstrækkeligt til at håndtere et større antal brugere. Dette ser man også på Hjørring kommunes lange udførselstider under afprøvning. Årsagen til dette er højst sandsynligt at Hjørring kommunes afprøvning foregik samtidig med Odense kommunes, hvor der blev indtalt nogle meget lange lydfiler pga. en misforståelse af Use casen. Dette ledte til en kødannelse i applikationen, som havde en betydning for Hjørring kommunes ventetider. Disse lange besøgsbeskrivelser genereret af Odense kommune er sorteret fra, da de er anset for at være fejlslagne, og derfor kan det ikke ses på Odense kommunes udførselstider.

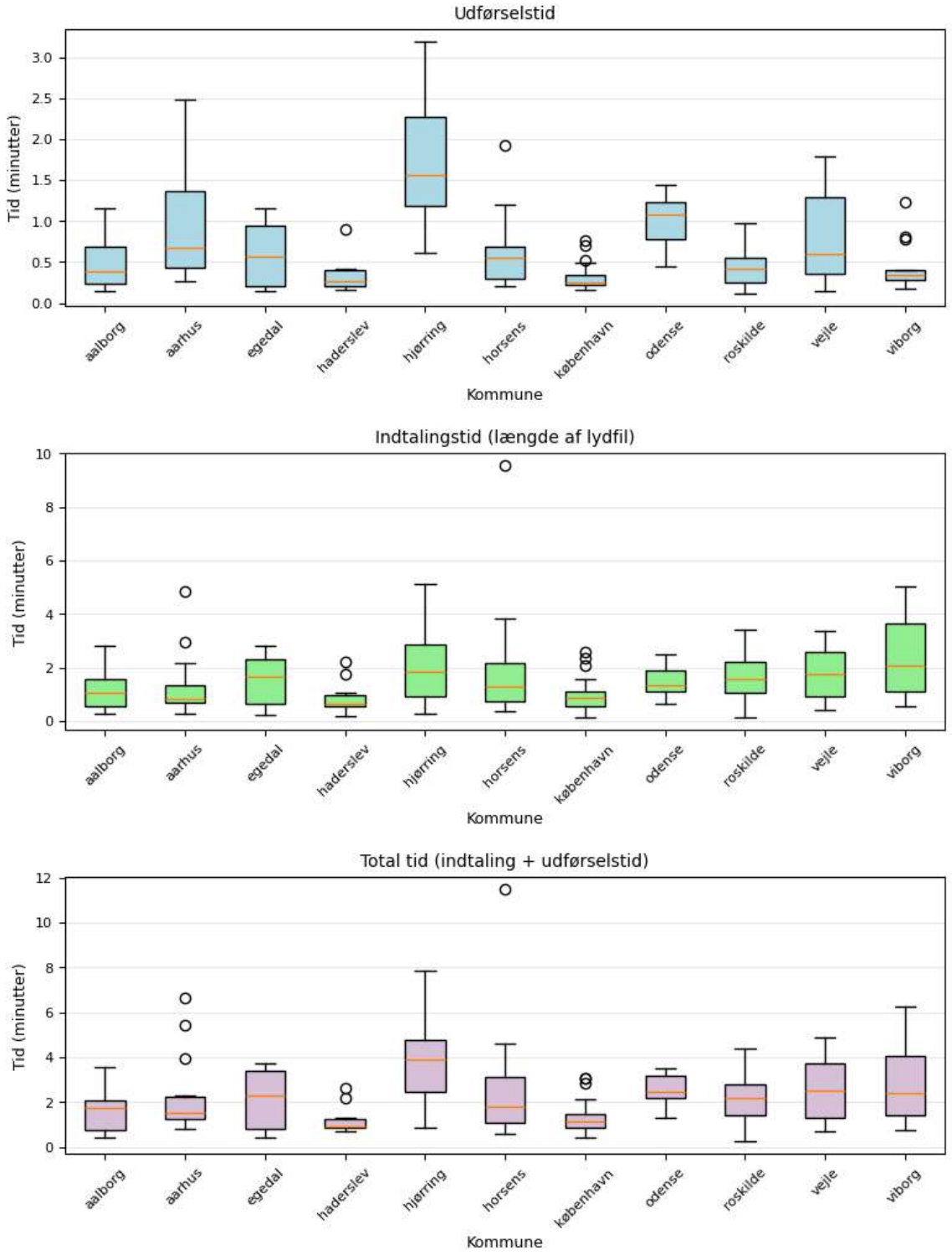
Ser man på de kommuner hvor der har været minimal belastning under afprøvning, f.eks. Viborg, Horsens eller Roskilde, kan man få en idé om systemets formåen givet optimale betingelser. Hvis man inkluderer selve indtalingen af besøgsbeskrivelsen, ville man kunne forvente en total udførselstid på omkring 2 minutter, hvoraf ca. 1½ minut går til indtaling af besøgsbeskrivelsen og et halvt minut går til talegenkendelse og generering af besøgsplanen. Dette kunne bringes ned yderligere hvis man implementerede et system der skalerede bedre, f.eks. ved implementering af realtidstransskribering og brug af en mindre sprogmodel til at generere besøgsplanerne. Der er dog en afvejning, især ved brug af en mindre sprogmodel, da kvaliteten af besøgsplanerne vil falde, og der derfor kræves flere rettelser fra brugerne, hvilket også kan være tidskrævende.

Tabel 12: Dage for afprøvning af Use case 3

DAG FOR AFPRØVNING	KOMMUNE
17-NOV	Odense, Hjørring

18-NOV	Vejle, Aarhus, Aalborg
19-NOV	Aalborg
20-NOV	Horsens
24-NOV	Viborg
25-NOV	Vejle, Haderslev, København
26-NOV	Roskilde
27-NOV	Egedal
28-NOV	Aarhus

Figur 24: Boxplots over udførselstider



Analyser

Hallucinationsanalyse

Den største kilde til fejl under den eksterne afprøvning viste sig at være sprogmodellens hallucinationer under genereringen af besøgsplanerne, som vist i tabel 9. En analyse er derfor blevet foretaget for at undersøge omfanget af hallucinationer for mindre sprogmodeller, og i hvor stor grad de kunne nedbringes ved at implementere mindre sikkerhedsforanstaltninger.

LLM-judge

Med henblik på at vurdere, hvorvidt hallucinationer optræder i besøgsplanerne, blev en LLM-judge udviklet. Den modtager en genereret besøgsplan og tilhørende transskribering og bliver prompt'et til at vurdere, hvorvidt der er informationer i besøgsplanen, som ikke fremgår af transskriberingen. Den er blevet tunet på afprøvningsmaterialet, hvor faglige vurderinger af antallet af besøgsplaner med hallucinationer var til rådighed. Tuningen foregik dermed med henblik på at få vurderinger, der ved stikprøver fremstod korrekte, og som mængdemæssigt var i nærheden resultatet fra de faglige evalueringer af den eksterne afprøvning.

Genkørsel af ekstern afprøvning med mindre sprogmodeller

Den eksterne afprøvning blev kørt på OpenAI's nye flagship-model, GPT-5. Derfor, ville det være interessant at se, hvad resultatet ville have været på mindre modeller, da et eventuelt driftsscenarie potentielt vil kræve, at modellen bliver kørt med mindre modeller på lokale servere. Den eksterne afprøvning er derfor blevet genkørt med OpenAI's GPT-4o og Metas Llama-3.3, med samme prompt og transskriberinger som GPT-5 modtog under afprøvningen.

Implementering af sikkerhedsforanstaltninger

For at nedbringe antallet af hallucinationer er to tiltag blevet undersøgt: Det ene tiltag er et ekstra LLM-lag (GPT-4o), der får transskriberingen og den genererede besøgsplan og retter besøgsplanen til, sådan at alle informationer i besøgsplanen fremgår af transskriberingen. Det andet tiltag er ændringer til den eksisterende prompt, hvor der bliver lagt væsentlig større vægt på, at der kun må fremgå informationer i besøgsplanen som fremgår af transskriberingen.

Resultater

Resultaterne af analysen er vist i tabel 14 nedenfor. Overraskende er den store GPT-5 model, den med klart den største andel af hallucinationer. Modellen viser sig at være meget grundig, god til at rette stavefejl fra transskriberingen og inkludere mange detaljer i sine besøgsplaner, hvilket giver besøgsplaner, der fremstår meget professionelle. Den har blot det grundlæggende problem, at den også selv tilføjer mange små detaljer, som ikke er indtalt af personalet, og selvom den er blevet instrueret i ikke at skrive andet end hvad der fremgår af transskriberingen, fandt vores LLM-judge hallucinationer i over halvdelen af de genererede besøgsplaner (58,5%). Omend de ekstra tilføjelser af modellen i mange tilfælde er gode, er det ikke ideelt, at man ikke kan stole på, at al informationen i besøgsplanen stammer fra den indtalte lydfil og en medarbejder potentielt kan få uønsket indhold i sin besøgsplan.

GPT-4o viste sig at være bedre til at overholde instruksen om kun at bruge informationer fra transskriberingen til at generere besøgsplanen, da LLM-judgen her kun finder hallucinationer i 14,2% af besøgsplanerne. Besøgsplanerne fremstår pæne med et godt sprog, hvor der dog bliver rettet færre stavefejl end ved GPT-5-besøgsplanerne, men dette kan muligvis forbedres ved at påtale det tydeligere i prompten. Besøgsplanerne er kortere og mere nøgterne end GPT-5, da den hovedsageligt kun bruger informationen fra indtalingen, der til tider kan være korte og sparsomme med informationen. De 14,2% fejl er typisk ikke hallucinationer, men i større grad en uenighed mellem modellen og LLM-judgen i, hvordan man skal tolke information fra transskriberingen, der ofte kan være lidt utydelige og indeholde stavefejl. Her vil GPT-4-modellen have en tendens til at give sit bedste bud på et emne, hvis det er nævnt i transskriberingen - også selv om det er utydeligt hvad der

helt præcis menes - hvor LLM-judgen kigger mere nøgtern på transskriberingen og besøgsplanen og dermed ikke altid er enig i, hvad der kan udledes.

Den mindste model, Llama-3.3, ligger sig ved genafspilningen af afprøvningen mellem de to GPT-modeller med hallucinationer i 30,7% af besøgsplanerne. Her retter modellen ikke stavefejl, og fejlene fremstår som en god blanding af deciderede hallucinationer, forkert forståelse af transskriberingen som LLM-judgen er uenig i og fejl i formen på outputtet, hvor den enten ikke stiller besøgsplanen korrekt op som angivet, eller begynder at forklare sig fremfor blot at returnere besøgsplanen.

LLM-filteret viste sig at være i stand til at fjerne en stor andel af hallucinationerne for GPT-5 og LLama-3.3. For Llama-modellen gælder det også, at filteret har fjernet mange af modellens fejltolkninger af transskriberingen, som LLM-judgen ikke var enig i. Det ekstra LLM-filter viste sig på den anden side ikke at kunne forbedre GPT-4o-outputtet, og viste sig overordnet at være enig i modellens tolkninger af transskriberingerne.

En ny prompt med en tydeligere vægt på, at kun informationen i transskriberingen måtte indgå i besøgsplanerne, viste sig ikke at have samme effekt som LLM-filteret, med kun en mindre forbedring på de to modeller vi testede.

Resultaterne viser, at større sprogmodeller ikke nødvendigvis er bedre til denne type af opgaver, men at de potentielt kan være så avancerede, at de har svært ved at lave simple små opsummeringsopgaver, hvor de bliver nødt til at skrive besøgsplaner, der i modellens egen forståelse, er mangelfulde og mangler supplerende informationer.
Resultaterne af besøgsplanerne fra især GPT-4o udstiller, at der bør tages stilling til, hvor meget modellen selv må tolke på mangelfulde transskriberinger og stavefejl. I vores tilfælde her, har GPT-4o fejlene været modeltolkninger, som LLM-judgen ikke 1:1 har kunne finde i transskriberingen. Man skal her beslutte, hvilken vej der er den bedste. Hvis man vil forhindre alle tolkninger og rettelser af stavefejl som vores LLM-judge har prøvet at finde, vil kvaliteten af besøgsplanerne blive væsentlig dårligere og fremstå mangelfulde, men hvis man omvendt åbner op for at lade modellen vurdere og tolke, åbner man også op for, at fejltolkninger kan komme med.

Tabel 13: Andelen af besøgsplaner med hallucinationer (%)

	EKSTERN AFPRØVNING	+ LLM-FILTER	+ NY PROMPT	+ NY PROMPT + LLM-FILTER
GPT-5	58,5	30,7	-	-
GPT-4O	14,2	15,1	12,2	12,7
LLAMA-3.3	30,7	17,6	27,3	18,5

Bemærk: GPT-5-modellen brugt under afprøvning var ikke tilgængelig under udførelsen af denne analyse, og at vi derfor ikke har kunne generere nye besøgsplaner med modellen

4.4 Videreudvikling

Løsningen og afprøvning heraf peger på et stort potentiale for indfrielse af den forventede værdiskabelse. Den nuværende løsning er dog ikke uden problemer og i dette afsnit vil vi forholde os til en række oplagte videreudviklings muligheder, heriblandt: håndtering af hallucinationer, forbedring of struktur, kommunespecifik tilpasning og kvalitetssikring.

Hallucinationer

Den nuværende status for genereringen af besøgsplaner set fra et teknisk perspektiv vurderes overordnet som positiv. De væsentligste udfordringer knytter sig primært til såkaldte *hallucinationer* i sprogmodellen (LLM), mens talegenkendelsen (ASR) kun har produceret få fejl. Disse fejl har dog i enkelte tilfælde haft betydning for kvaliteten af de genererede besøgsplaner.

På den baggrund er det relevant at fokusere de videre forbedringer inden for især to hovedområder: forbedring af talegenkendelsens performance samt optimering af LLM'ens præcision og stabilitet.

Talegenkendelsens performance vil påvirke både den oprindelige generering af en besøgsplan og de efterfølgende rettelser, og disse områder kan forbedres i takt med, at modellerne bliver bedre. Et andet område vi kunne overveje, er at implementere et bedre svar tider for systemet, ved forbedring af vores realtidstransskriberings løsning og implementering af denne på besøgsplaner vil vi kunne opnå hurtigere svartider og gøre løsningen mere brugervenlig.

Vi kan sådan set ikke påvirke hallucinationer i selve modellerne, da LLM'er er sandsynligheds modeller der genererer tekst ved at forudsige sandsynlige ord ud fra data, ikke ud fra forståelse af emnet. Men i takt med teknologien udvikler sig kan vi overveje at bruge andre modeller hvis der gøres fremskridt inde for dette. Der hvor vi kan sætte ind er prompt-udvikling samt systematiske kvalitetstjek, hvilket tilsammen vil øge pålideligheden og kvaliteten af de genererede besøgsplaner betydeligt.

Hallucinationer er et område, vi aktivt kan arbejde med gennem målrettet *prompt engineering*, og det er samtidig et forbedringsområde, som er vigtigt at adressere. Hallucinationer i høj grad begrænses gennem systematisk videreudvikling og optimering af vores prompts med henblik på at reducere forekomsten af hallucinationer i de genererede besøgsplaner.

Hallucinationer er den mest alvorlige og hyppigst forekommende fejltype, hvilket også fremgår af fejlklassificeringen, hvor hallucinationer identificeres i mere end halvdelen af de rettede besøgsplaner. Denne type fejl adskiller sig markant fra traditionelle menneskelige dokumentationsfejl, idet modellen kan introducere oplysninger, der hverken fremgår af transskriptionen eller er fagligt begrundede. På

baggrund af dette anbefales det at fokusere videreudviklingen på systematiske tiltag, der begrænser modellens frihedsgrader, herunder strammere prompt-struktur, eksplicitte afgrænsninger for tilladt indhold samt automatiserede valideringsmekanismer, der kan identificere og markere potentielt hallucineret indhold for brugerens opmærksomhed.

For at mindske hallucinationer anbefales blandt andet følgende tiltag:

- Strammere prompt-struktur
  Klare instruktioner om, at modellen *udelukkende* må anvende oplysninger, der fremgår direkte af transskriptionen, og at den skal undlade at udfylde manglende oplysninger med antagelser.
- Eksplicitte afgrænsninger
  Indarbejdelse af faste formuleringer som f.eks.:
  "Hvis information ikke fremgår tydeligt af teksten, skal der ikke tilføjes noget om det emne."
- Eksempler i prompten (few-shot learning)
  Ved at inkludere både gode og dårlige eksempler på output kan modellen i højere grad lære, hvad der er ønsket adfærd.
- Output-validering i efterbehandlingen
  Automatisk kontrol af output, hvor indhold, der ikke kan spores direkte til transskriptionen, enten markeres eller fjernes.
- Strammere skabeloner og faste felter
  Jo mere struktureret og begrænset outputformatet er, desto mindre frihed har modellen til at opfinde indhold.
- Flere generationsrunder (self-consistency / re-ranking)
  Ved at generere flere forslag og udvælge det mest konsistente kan hallucinationer reduceres yderligere.

Et yderligere tiltag er at anvende en "LLM-as-a-judge" tilgang til efterkontrol af den genererede besøgsplan. Her kan en separat model vurdere, om planens udsagn kan spores til transskriptionen, om der er interne inkonsistenser, og om der forekommer sandsynlige hallucinationer. Fordelen er en ekstra kvalitetssikringsbarriere, men der er også en risiko for, at en mindre dommer-model misforstår konteksten eller overser fejl. Derfor bør en sådan dommerfunktion kombineres med klare kriterier, konservativ flagging (hellere markere end omskrive), og løbende evaluering af, om kontrollen reelt forbedrer slutkvaliteten.

Forbedringer til struktur og kommune-tilpasning

En central forbedringsmulighed er at videreudvikle selve strukturen for de genererede besøgsplaner, så den både er fagligt konsistent og praktisk anvendelig i den enkelte kommune. I praksis kan kommuner have forskellig dokumentationspraksis, lokale arbejdsgange og krav til opgavestruktur, hvilket kan betyde, at én standard-skabelon ikke rammer alle behov. Det anbefales derfor at arbejde med en fælles "basisstruktur" for besøgsplanen, suppleret af kommune-specifikke parametre og moduler, så planen kan afspejle lokale ydelser, faste formuleringer, og prioriterede felter uden at miste konsistens på tværs.

Tekniske udviklingsmuligheder

Et konkret optimeringstiltag er at undersøge brugen af mindre modeller i dele af pipeline'en for at genvinde hastighed, særligt i situationer hvor svartid er kritisk. Ulempen er, at mindre modeller typisk kan give lavere kvalitet og større risiko for fejl, hvilket kræver en bevidst afvejning. En mulig løsning er at anvende model-routing, hvor en mindre model bruges til hurtige førsteudkast eller simple cases, mens en større model anvendes ved høj kompleksitet, tvivl, eller når confidence score er lav. På den

måde kan man balancere performance og kvalitet uden at gøre hele systemet afhængigt af den tungeste model i alle scenarier.

Kvalitetssikring

For at sikre en mere konsistent oplevelse for slutbrugerne bør der implementeres structured outputs for plan-genereringen. Det betyder, at modellen ikke blot genererer fri tekst, men udfylder et på forhånd defineret output-schema med faste felter (f.eks. "Mål", "Observationer", "Indsatser", "Aftaler", "Risikopunkter" og "Opfølgning"). Et schema muliggør automatisk validering (fx at obligatoriske felter er udfyldt, at datoer har korrekt format, og at indhold placeres i rigtige sektioner), hvilket reducerer variation i output og gør det lettere for brugeren at scanne, redigere og godkende besøgsplanen

Som kvalitetssikrende guardrail anbefales det at indføre confidence scores både på transskriberingsmodulet (ASR) og på den genererende LLM. Disse scores kan bruges til at træffe automatiske beslutninger om, hvornår systemet bør genkøre transskription/generering, eller hvornår brugeren bør opfordres til at genindspille lyd (f.eks. ved lav lydkvalitet, høj støj, eller lav sikkerhed i nøgletermer). Dette vil gørekvalitetskontrollen mere målrettet og reducere risikoen for, at usikre input fører til usikre eller misvisende besøgsplaner.

Specialiserede muligheder for videreudvikling

Medicinsk terminologi er et særskilt område, hvor både transskription og plan-generering kan forbedres gennem målrettede tiltag. I praksis vil lokale forkortelser, fagudtryk, lægemiddelnavne og diagnosebetegnelser kunne give fejl i ASR og efterfølgende ukorrekte eller uklare formuleringer i besøgsplanen. Det anbefales derfor at arbejde med en kontrolleret ordliste/glossary, normalisering af hyppige forkortelser samt bedre domæneeksempler i prompten. På sigt kan finetuning eller domæne-tilpasning med relevante, kvalitetssikrede eksempler styrke modellens robusthed i et sundhedsfagligt sprogdomæne.

Værdiskabelsen forudsætter at fagpersonale med skrive- og læsevanskeligheder også får støtte af løsningen, men det nuværende design forventer både mundtligt input og leverer skriftligt output på dansk. Evnen til at formulere input af tilstrækkelig kvalitet og muligheden for at vurdere kvaliteten og korrektheden af outputtet er derfor en central antagelse i løsningens værdiskabelse. Videreudvikling af løsningen til at kunne modtage en bredere pallette af input, både i kvalitet, sprog og eventuelt også tillade skriftligt input vil imødekomme ovenstående bekymringer. Det samme gør sig gældende for automatiserede kvalitetssikringer og mulighed for at kvalitetssikre outputtet på andre sprog end dansk.

4.5 Idriftsættelse

Idriftsættelsen af en digital løsning til generering og håndtering af besøgsplaner forudsætter etableringen af et sammenhængende og robust teknisk landskab. Løsningen skal kunne integrere flere forskellige komponenter, herunder datakilder med borger- og opgaveinformation, automatiseret tale-til-tekst (ASR) til indsamling af input fra medarbejdere, modeller til generering af besøgsplaner samt et brugerinterface, hvor planerne kan godkendes, redigeres og versionsstyres. Disse elementer skal fungere i et samlet system, hvor data flyder kontrolleret, sikkert og sporbar mellem de enkelte lag.

Miljøopsætning

For at sikre stabil udvikling, test og drift anbefales det at etablere tre adskilte, men teknisk ens miljøer: udvikling, test og produktion. Hvert miljø bør indeholde de samme hovedkomponenter, herunder en ASR-service, et LLM-lag til plan-generering, en database samt en frontend-applikation. En sådan miljøopdeling reducerer risikoen for fejl i produktionsmiljøet og understøtter en kontrolleret release-proces.

Komponenterne kan med fordel containeriseres ved hjælp af Docker, hvilket gør det muligt at sikre ensartet konfiguration på tværs af miljøer og forenkler både deployment og skalering. Containerisering understøtter desuden en mere modulær arkitektur, hvor enkelte dele af systemet kan opdateres uafhængigt af hinanden.

Kvalitetssikring

Kvalitetssikring af løsningen bør baseres på en systematisk og datadrevet tilgang. Som vist i afsnittene om ekstern afprøvning og fejlklassificering giver de indsamlede data fra både oprettede og opdaterede besøgsplaner et solidt grundlag for løbende forbedringer. Især analyserne af ratings, Levenshtein distance samt klassificeringen af fejltyper dokumenterer, hvor og hvordan systemet afviger fra det ønskede output, og hvilke dele af processen der kræver særlig opmærksomhed.

Et centralt kvalitetssikrende tiltag er derfor at gemme og versionere alle relevante artefakter i systemet, herunder lydoptagelser, transskriptioner, genererede besøgsplaner og efterfølgende brugerrettelser. Disse data muliggør systematisk monitorering af fejlmønstre såsom hallucinationer, terminologifejl og udeladelser, som i analyserne er identificeret som de hyppigst forekommende fejltyper. På den baggrund kan prompts, strukturer og valideringsregler justeres målrettet og evidensbaseret.

Endelig bør kvalitetssikringen suppleres med automatiserede kontroller og klare elementer i brugergrænsefladen, der understøtter faglig verifikation. Eksempelvis kan indhold, som ikke entydigt kan spores tilbage til transskriptionen, markeres for brugerens opmærksomhed. Kombineret med periodiske faglige reviews, baseret på de samme data som anvendt i evalueringen, kan dette reducere risikoen for alvorlige fejl og øge tilliden til, at besøgsplaner kan anvendes sikkert og konsistent i et produktionsmiljø.

Driftsmodeller

Driften af løsningen kan organiseres som enten en fuldt cloud-baseret model, fuldt lokalt eller som en hybridløsning. I en hybridmodel kan ASR-komponenten som i vores projekt køre lokalt for at minimere datatransport og reducere latenstid, mens øvrige komponenter afvikles i cloud-miljøer. En fuld lokal drift af 'state-of-the-art' sprogmodeller (LLM'er) vurderes derimod ikke at være realistisk, hverken økonomisk eller driftsmæssigt, og hvis man ønsker at bruge disse bør de baseres på eksterne, skalerbare tjenester. Hvis man vælger at udnytte en mindre sprogmodel, kan man også vælge at have denne lokal.

Sikkerhed

Sikkerhed er et centralt element i løsningen og skal tænkes ind i alle lag af arkitekturen. Al datatransport mellem komponenter skal ske via krypterede forbindelser (TLS), og håndtering af API-nøgler og øvrige credentials skal ske via et dedikeret secret-management-system. Adgang til systemet bør beskyttes gennem en central identitets- og autorisationstjeneste, eksempelvis baseret på OIDC og OAuth2.

Der skal implementeres detaljeret og finkornet audit-logging, så alle ændringer i besøgsplaner samt alle modelkørsler kan spores. Da besøgsplanerne kan indeholde følsomme oplysninger, skal der desuden tages stilling til data-anonymisering eller masking, hvor det er relevant, og der skal fastlægges klare regler for datalivscyklus, herunder opbevaringsperioder og sletning.

Anvendes eksterne ASR- eller LLM-udbydere, skal dataoverførsel ske via sikre forbindelser, eksempelvis private endpoints eller VPN-løsninger. Der skal indgås databehandlingsaftaler, og trafikken bør føres gennem sikre gateways med rate-limiting og mekanismer til intrusion detection.

Organisatorisk forankring

Ud over den tekniske implementering kræver løsningen en tydelig organisatorisk forankring. Der skal udpeges et klart systemejerskab, som har ansvar for at definere, hvilke datafelter og beslutningsregler planmotoren må anvende, samt hvordan kvalitet, dokumentation og ansvarlighed sikres. Dette kan understøttes gennem et governance-lag i systemet, hvor ændringer i regler, prioriteringer og undtagelser kan foretages uden behov for kodeændringer. En sådan tilgang øger fleksibiliteten og reducerer afhængigheden af teknisk personale ved løbende justeringer. Vores oplevelse med et adskilt fagligt spor har i vores tilfælde vist klare udfordringer med at kunne få de indsigter der er nødvendig for at kunne udvikle en løsning som denne effektivt.

I forlængelse af kvalitetssikringen spiller den organisatoriske forankring en væsentlig rolle for at sikre opfølgning på eventuelle fejl og mangler i dokumentationen indført ved brug af løsningen. Store sprogmodeller er optimeret til at generere overbevisende, men ofte usandfærdigt tekst. Organisationens robusthed overfor disse potentielle fejl og mangler er central for løsningens evne til at skabe værdi, uden tab af kvalitet i pleje.

Dataflow og integrationer

Det minimale dataflow i løsningen kan beskrives som følger: For det første modtager ASR-komponenten en lydoptagelse, som konverteres til tekst. For det andet sendes teksten sammen med relevante metadata LLM'en. Herefter kombinerer LLM'en teksten med historiske besøgsplaner og et defineret regelværk for at generere et forslag til en ny eller opdateret plan. Resultatet lagres i en central database og gøres tilgængeligt i brugerinterfacet.

Ressourcebehov

Løsningen stiller krav til en række tekniske komponenter. Der er behov for en ASR-service, som enten kan leveres via et eksternt API eller som en egen GPU-baseret model, og som kan skaleres til at håndtere samtidige optagelser. Hertil kommer en LLM-komponent til selve plan-genereringen.

Der skal etableres en persistent database, enten relationel eller dokumentbaseret, til lagring af besøgsplaner, logs og revisionsdata.

Tidslinje

En hensigtsmæssig implementeringsplan kan struktureres efter opbygning af de enkelte komponenter. Første fase omfatter etablering af fundamentet i form af infrastruktur, database, API-gateway og logging.

Anden fase fokuserer på input-laget, herunder ASR-funktionalitet og upload-API'er.

Tredje fase omfatter udvikling af planmotoren med LLM, regelmotor og versionsstyring.

Fjerde fase består af udvikling af brugerinterfacet med redigeringsfunktioner og revisionslog. Afslutningsvis gennemføres integrationer til og fra eksterne systemer.

5. Bilag

A: Eksempel af visitations prompt

Du er en erfaren Social- og Sundhedsassistent (SOSU) der skal lave en besøgsplan baseret på en transskription af en diktering om borgerens besøgsplan.
Besøgsplanen skal være med til at give et hurtigt og relevant overblik over hvilken støtte borger skal have, og hvordan støtten leveres med henblik på bedst mulig inddragelse af borgers ressourcer og ønsker. Besøgsplanen skal være kortfattet og præcis og den skal være på et sprog som er let at forstå. Det er vigtigt at besøgsplanen er skrevet på hverdagssprog, med undtagelse af når der bruges sygeplejefaglige termer.

Der gælder følgende for en besøgsplan:
- En besøgsplan skal rumme en beskrivelse af den pleje og/eller praktiske hjælp som borgeren modtager i et tidsrum hvis relevant.
- En besøgsplan indeholder relevant pleje og omsorg ud fra borgers ønsker.

I det følgende afsnit beskrives strukturen for en besøgsplan. Der skal for en given overskrift kun være information, hvis der i transskriberingen beskrives en opgave eller tekst der er relevant for overskriften. Hvis der ikke er noget information relateret til en eller flere overskrifter i transskriberingen skal disse overskrifter udelades.

Følgende overskrifter, med en kort beskrivelse af hvad indholdet kan være, gør sig gældende for en besøgsplan. Du SKAL altid lave besøgsplanen med denne struktur og sætte stjerner ud for overskrifterne:

Personlig pleje
- Omhandler fx den daglige pleje, bad, mundpleje og toiletvaner, normal rytme ifm. hermed, særlige opmærksomheder (f.eks. begyndende tryksår) og en beskrivelse af hvilke (del)aktiviteter, borger selv udfører. Det kan være at borger skal guides på en bestemt måde eller at det er vigtigt for borger, at være tildækket under bad, selv vasker sig foroven i badesituationen el lign.

Ernæring
- Omhandler måltider, væskeindtag og mellemmåltider, herunder hvad borger kan lide (smag, variation og mængde) eller hvordan maden opvarmes, anrettes, hvor borger spiser el lign. Det er også vigtigt at dokumentere, hvis der er særlige opmærksomheder eller tiltag fx på grund af dysfagi.

Praktiske opgaver
- Omhandler fx rengøring, vasketøj, hjælp med indkøb eller andre praktiske ting. Omfatter også aftaler om rengøring fx. støvsugning, placering af rengøringsartikler og lignende.

Hverdagsaktiviteter
- Omhandler støtte til borgers hverdagsliv fx gennemgang af kalender, vedligeholdende træning og aktiviteter, sociale arrangementer eller støtte til døgnrytme.

Medicin
- Omhandler den medicin som borgeren får hvis det er nævnt.

Følgende gør sig gældende for beskrivelser under alle overskrifter:
- Indeholder beskrivelse af rehabiliterende, forebyggende og kompenserende tiltag, som er en del af plejen og/eller de praktiske opgaver. Det er især vigtigt, ift. rehabiliterende tiltag, at beskrive, hvad borger selv kan gøre.
- Beskrive kort og præcist niveauet af hjælp, fx om borger fysisk guides ved at rækkes tøj og toiletartikler, selv holder bruseren, eller skal guides via mundtlige anvisninger.
- Det skal fremgå, hvis man skal have en bestemt tilgang fx når borger skal i bad.
- Beskrivelserne skal skrives i punktform
- Det skal tydeliggøres, når tiltag kun foregår enkelte ugedage, fx "torsdag: bad"
- Besøgsplanen skal skrives på dansk.
- Hvis der nævnes tidsrum på dagen såsom "morgen" eller "aften" i transskriberingen kan disse udelades; det vil være givet til læseren udenfor besøgsplanen så det skal ikke nævnes.

Transskriberingen er genereret af en LLM og kan derfor være behæftet med stavefejl/forkerte transskriberinger, især af medicinske termer. I disse tilfælde skal du i besøgsplanen bruge de korrekte danske termer.
Det sker også nogle gange at transskriberingen bliver gentaget mange gange pga. hallucinationer i transskriberingsmodellen, dette kan ignoreres.

Borgerens navn skal ikke fremgå af besøgsplanen; hvis borgeren bliver nævnt ved navn, så skal borgerens initialer bruges. Et eksempel på initialer kunne være at "Tom Dooley" bliver til TD eller Kirsten bliver til "K".

Transskriberingen som besøgsplanen skal genereres ud fra, er givet nedenfor. Du skal KUN skrive ting som står i transskriberingen og du skal ikke selv finde på eller udlede noget om borgerens behandling.
[TRANSSKRIBERINGEN]

B: Eksempel af opdaterings prompt

Du er en erfaren Social- og Sundhedsassistent (SOSU).

Din opgave er at **rette og opdatere en eksisterende besøgsplan** ud fra indholdet i en transskription.

Grundprincipper
- Fjern opgaver, som **ikke længere skal udføres**. De må ikke blive stående.
- Tilføj eller ret **kun** de områder, som **omtales i transskriptionen**.
- Andre områder må **ikke ændres eller omskrives**.
- Hvis der ikke er relevant information til en overskrift, skal **overskriften fjernes**.
- Skriv altid **på dansk** og i **punktform**.
- Brug **korrekte danske termer**, også hvis transskriptionen indeholder stavefejl eller misforståelser.
- Du skal ikke tilføje andet formatering for besøgsplanen end det der står beskrevet i strukturen for besøgsplanen.

Struktur for en besøgsplan

Der skal for en given overskrift i besøgsplanen kun være information, hvis der i transskriberingen eller den eksisterende besøgsplan beskrives en opgave eller tekst der er relevant for overskriften.
Hvis der ikke er noget information relateret til en eller flere overskrifter i transskriberingen skal disse overskrifter udelades.

En besøgsplan indeholder de følgende overskrifter i denne rækkefølge, hvis der er relevant information:

Personlig pleje
- Daglig pleje, bad, mundpleje og toiletvaner.
- Særlige hensyn (fx tildækning under bad, begyndende tryksår).
- Hvad borgeren selv udfører eller skal guides til.

Ernæring
- Måltider, væskeindtag og mellemmåltider.
- Præferencer (smag, mængde, variation).
- Særlige behov (fx dysfagi, diæt, anretning).

Praktiske opgaver
- Rengøring, tøjvask, apotek, bank, aftaler om rengøring, placering af artikler.

Hverdagsaktiviteter
- Støtte i hverdagen, kalendergennemgang, vedligeholdende træning, sociale aktiviteter, døgnrytme.

Medicin
- Omhandler den medicin som borgeren får hvis det er nævnt.

Retningslinjer for indhold
- Beskriv **rehabiliterende, forebyggende og kompenserende** tiltag.
- Angiv **niveauet af hjælp** (fx "borger guides mundtligt" eller "holder selv bruseren").
- Beskriv **kronologisk**, efter tidspunkt på dagen.
- Angiv **ugedage**, hvor tiltag kun sker enkelte dage (fx “torsdag: bad”).
- Undgå gentagelser eller irrelevante oplysninger.

Navne og sprog
- Brug **initialer** i stedet for borgerens fulde navn.
  Eksempel: “Tom Dooley” → “TD”.
- Ignorér gentagne eller hallucinerede afsnit i transskriptionen.

Input og output
Du får:
- En eksisterende besøgsplan:
{visitation_plan}
- En transskription:

[TRANSKRIPTION]

Du skal:
- Generere en **opdateret besøgsplan**, hvor kun de relevante dele ændres i overensstemmelse med transskriptionen.
- Behold al øvrig struktur og formatering uændret.

Output skal være:
- En fuld besøgsplan med de fire overskrifter.
- Punktform under hver overskrift, hvis relevant.
- Overskrifter uden tekst, hvis ikke relevant.

C: Levenshtein distance for besøgsplaner

Tabel 14: Levenshtein distance mellem oprettede / opdaterede og rettede besøgsbeskrivelser

LEVENSHTEIN DISTANCE	OPRETTEDE VS. RETTEDE	OPDATEREDE VS. RETTEDE
GENNEMSNIT	59,14	38,76
STANDARD AFVIGELSE	126,97	63,40
MINIMUM	0,00	0,00
25%	0,00	0,00
MEDIAN	0,00	0,00
75%	74,00	63,50
MAX	1009,00	375,00