UNIVERSITETET I AGDER

Forside

IS-304: 2024

Tittel: QuickFix – Støtteverktøy for kundeservice i Sikri

Emnekode	IS-304
Emnenavn	Bacheloroppgave i informasjonssystemer
Emneansvarlig:	Geir Inge Hausvik - Hallgeir Nilsen
Veileder	Jaziar Radianti
Oppdragsgiver:	Sikri

Studenter:

Etternavn	Fornavn
Demirezen	Samet
Seymen	Burak
Yildirim	Orhan
Yildirim	Umit
Yilmaz	Busenur

Jeg/vi bekrefter at vi ikke siterer eller på annen måte bruker andres arbeider uten at dette er oppgitt, og at alle referanser er oppgitt i litteraturlisten.	Ja X	
Kan besvarelsen brukes til undervisningsformål?	Ja X	
Vi bekrefter at alle i gruppa har bidratt til besvarelsen	Ja X	

Forord

Vi vil gjerne uttrykke vår dypeste takknemlighet til alle i Sikri og alle andre som har veiledet og støttet oss gjennom fullføringen av dette prosjektet. Vi ønsker spesielt å takke vår veileder ved Sikri, Kristoffer Stokkeland. Kristoffer har gitt oss omfattende støtte gjennom alle stadier av prosjektet, organiserte ukentlige møter og svarte raskt på meldinger uansett tid på døgnet, og engasjerte seg dypt i vårt arbeid. Vår takknemlighet overfor Kristoffer er umåtelig, da han har inspirert og motivert oss gjennom hele prosjektet.

Vi vil også takke Bjørnar Sømme hos Sikri, som har vært til stor hjelp. Bjørnar viste vårt arbeidsområde og sørget for at vi hadde alt vi trengte for å bidra til prosjektet, og han gjorde sitt ytterste for at vi skulle føle oss som en del av fellesskapet ved å invitere oss til alle organisasjonsarrangementer, noe som hjalp oss å utvide vårt sosiale og profesjonelle nettverk.

Til slutt vil vi takke Hallgeir Nilsen og Geir Inge Hausvik for all akademisk støtte og oppmuntring. Vi vil også gjerne takke Jaziar Radianti spesielt for hennes veiledning gjennom hele prosessen og under skrivingen av rapporten. Jaziars råd har styrket de vitenskapelige og akademiske aspektene ved prosjektet vårt. Vi har hatt en kontinuerlig og effektiv kommunikasjon, og Jaziar har ikke bare gitt nyttige tilbakemeldinger, men også spilt en stor rolle i å holde motivasjonen vår høy. Vi er veldig takknemlige for henne.

Gjennom arbeidet måtte vi fortløpende tilegne oss ny kunnskap, og vi møtte noen utfordringer. Vi så på det å overvinne disse utfordringene som en del av arbeidet vårt, og som et team har vi fått verdifull kunnskap og erfaringer som vi vil dra nytte av i arbeidslivet. Erfaringene og lærdommene vi har fått hos Sikri vil fortsette å veilede oss på vår karrierevei. Vi takker alle og håper på å opprettholde produktive samarbeid i fremtidige prosjekter.

Sammendrag

I dette prosjektet har vi utviklet en webapplikasjon for Sikri AS, en ledende norsk programvarebedrift, for å effektivisere håndteringen av kundehenvendelser. Prosjektet adresserer tregheten i den nåværende operasjonelle støtteprosessen ved å tilby et nytt, brukervennlig grensesnitt som lar kundeservicemedarbeidere håndtere enkle og kategoriserte saker direkte. Dette nye systemet vil øke hastigheten og kvaliteten på kundestøtten, redusere tidsbruk og kostnader, og forbedre kundetilfredsheten. Applikasjonen, utviklet gjennom en agil prosjektmetodikk og basert på React for frontend og ASP.NET for backend, støtter sikker oppdatering av brukerinformasjon gjennom JSON-filer og har strenge kvalitetssikringstiltak. Gjennom kontinuerlige sprints og samarbeid med Sikri, har vi implementert et system som ikke bare møter de tekniske behovene, men også forbedrer brukeropplevelsen. Prosjektet har benyttet seg av moderne programvareutviklingsverktøy og -praksiser som Azure DevOps og Git, sikret høy kvalitet og pålitelighet i produktleveransen. Gjennom denne prosessen har vi oppnådd en dypere forståelse av kundeservicens utfordringer og utviklet en løsning som er robust, fleksibel og tilpassbar for fremtidige behov. Denne oppgaven har ikke bare forbedret vår tekniske kompetanse, men også vår evne til å arbeide i team og løse komplekse problemer effektivt.

Her er video lenken til ferdig system:
https://drive.google.com/file/d/15IG3CeAKIvU3HJullHxr54R5jCdLPWjk/view?usp=sharing

Innholdsfortegnelse

Figurlist

Tabell

1. Introduksjon

I enhver organisasjon spiller kundestøtte en kritisk rolle, da den direkte påvirker kundetilfredsheten og selskapets rykte. Å gi svar som er i tråd med kundens forventninger, tilnærme seg deres situasjoner med empati og tilby effektive løsninger på problemene de møter, spiller en kritisk rolle i å øke kundetilfredsheten og bygge langsiktig kundelojalitet (Nyvoll, 2023). I en tid der digitalisering akselererer og kundenes forventninger stadig øker, blir behovet for å optimalisere kundestøtteprosesser enda mer presserende.

Prosjektet presentert her tar sikte på å adressere en vesentlig utfordring innen Sikris kundeservicetjenester – tregheten og effektiviteten i den nåværende operasjonelle støtteprosessen. Ved å utvikle en ny webapplikasjon som effektiviserer håndteringen av kundehenvendelser og klager, tar Sikri sikte på å forbedre både hastigheten og kvaliteten på kundestøtten. Dette vil ikke bare redusere tidsbruken og kostnadene forbundet med saksbehandlingen, men også forbedre kundetilfredsheten og styrke selskapets omdømme.

Det nye initiativet illustrerer Sikris forpliktelse til kontinuerlig forbedring og innovasjon, og understreker viktigheten av å ha effektive, sikre og fleksible systemer for å møte dagens digitale utfordringer. Ved å gi kundeservicemedarbeidere de verktøyene de trenger for å håndtere forespørsler direkte, viser Sikri en forståelse for viktigheten av kundestøtte som en integrert del av organisatorisk suksess og kundeopplevelse.

Dette prosjektet (QuickFix) har vist at kundeservice kan løse noen av de gjentakende problemene fra kunder uten å måtte overføre dem til en annen avdeling, og uten å skape mulige feil som kan forstyrre tjenesteflyten.

1.1 Sikri

Sikri AS er en av Norges ledende programvarebedrifter, med over tretti års historie (Sikri, u.å.). Selskapet er spesielt anerkjent for sitt samarbeid med både offentlig og privat sektor innen internkontroll, kvalitet, saksbehandling og arkiveringssystemer. Sikri samarbeider tett med nøkkelinstitusjoner som Kommunesektorens organisasjon (KS), Digitaliseringsdirektoratet, Direktoratet for Byggkvalitet og Arkivverket for å hjelpe den offentlige sektoren med å nå sine mål. Som en del av Spir Group fungerer Sikri i skjæringspunktet mellom offentlig og privat sektor, og spesialiserer seg på programvare og systemer som sikrer trygg datadeling og digitaliserte prosesser for kunder og partnere. Sikri samarbeider godt med akademia, og ressurser er blitt allokert av Sikri for å støtte gruppens arbeid i hver fase av bacheloroppgaven.

1.2 Oppgavebeskrivelse

Prosjektet utvikler et internt nettbasert verktøy for Sikri's kundesenter, designet for å øke hastigheten og sikkerheten ved håndtering av kundehenvendelser og klager. I det nåværende systemet blir alle henvendelser og klager sendt til operasjonell støtte (OPS), noe som kan forsinke

prosessen, føre til tidstap og økte kostnader. Den nye webapplikasjonen (QuickFix) som skal utvikles vil gi kundeservicepersonell et bærekraftig grensesnitt som lar dem håndtere enkle og kategoriserte saker direkte, og dermed spare tid og kostnader.

Målet med prosjektet er å digitalisere prosessene for kundeservice, og skape et system som er sikkert og kan kontrolleres. Systemet vil tillate kundeservicepersonell å foreta sikre endringer i lagrede JSON-filer som ligger på Sikris server på Azure DevOps uten å ha uautorisert tilgang til kildekoden. Det forventes at systemet vil være forbedringsdyktig og alltid ha effektive kvalitetssikringstiltak på plass.

Forventede fordeler inkluderer rask og feilfri oppdatering av brukerinformasjon, robusthet mot brukerfeil og automatiske kontroller for å forhindre uønskede sideeffekter. I tillegg er det et mål at det skal være enkelt å legge til nye funksjoner og oppdateringer, og at systemet kan tilpasse seg økende behov.

Som et resultat vil dette systemet øke effektiviteten til kundeservicemedarbeiderne og spare tid og ressurser. Effektivisering av kundeservicen vil føre til høyere kundetilfredshet. Programvarekvaliteten vil bli opprettholdt på et høyt nivå med automatiske tester og distribusjonsprosesser, og potensielle feil vil bli identifisert og løst på et tidlig stadium. Generelt sett er hovedmålet med prosjektet å sikre at kundenes forespørsler og klager blir løst så raskt som mulig, og å bidra til å forkorte prosessen.

1.3 Problemdomene

Prosjektet er designet for å forbedre effektiviteten til Sikris kundeservicetjenester. Problemdomene adresserer tregheten i den nåværende operasjonelle støtteprosessen (OPS) og behovet for å redusere kostnader og tid brukt på håndtering av kundeforespørsler og klager. Kundeservicemedarbeidernes begrensede evne til å løse ofte forekommende og repeterende problemer og avhengigheten av OPS for alle typer problemer utgjør hovedproblemene.

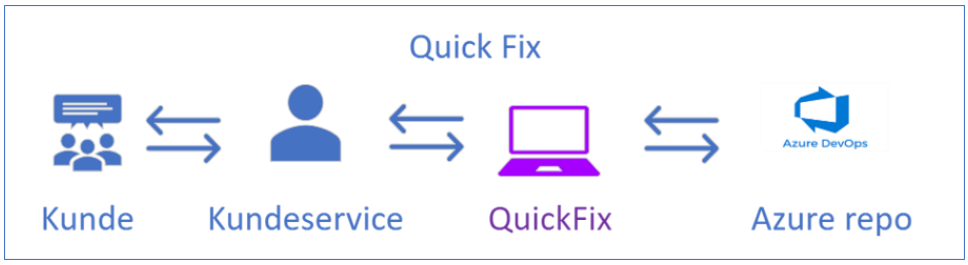
Figur 1 : Nåværende system

I den nåværende operasjonen registrerer kunder saker i kundeportalen på https://www.sikri.no/, hvorpå de får bekreftet registreringen og tildeling av en saksbehandler. Kundesenteret mottar deretter billetten og fordeler saken i henhold til tjenestenivåavtalen (SLA-Service Level Agreement), med prioriteringer fra lav til kritisk. Kundesenteret, delt inn i en 2. og 3. linje, sørger for at 2. linje filtrerer alle nye saker. Når vi snakker om "2. linje" og "3. linje" i konteksten av et

kundesenter, refererer vi til ulike nivåer av teknisk støtte eller kundestøtte. Disse nivåene er en del av en strukturert tilnærming til problemhåndtering og kundestøtte, hvor hver "linje" representerer et økende nivå av ekspertise eller spesialisering. For mindre komplekse henvendelser, som endring av publiseringstid, sendes en forespørsel til OPS via Teams for å utføre endringen. Mer tekniske eller komplekse saker sendes derimot til 3. linje, som har større teknisk kompetanse og tilgang til å kjøre databasespørringer ved behov, men også de må kontakte OPS for konfigurasjonsendringer. Når OPS har utført endringen og bekreftet dette, bekrefter kundesenteret endringen med kunden og lukker saken.

Dette systemet var problematisk fordi langsomheten i prosessen påvirker negativt kundetilfredsheten og kvaliteten på tjenestene. I tillegg fører overbelastningen av OPS til at medarbeiderne ikke kan fokusere på mer komplekse problemer, noe som reduserer den generelle effektiviteten. Utviklingen av den nye webapplikasjonen tar sikte på å løse enkle og gjentatte kundeforespørsler raskt og effektivt, slik at både kundetilfredshet og operasjonell effektivitet forbedres.

Gruppen setter søkelys på å utvikle en løsning som møter disse behovene og gir kundeservicepersonalet et sikkert, kontrollerbart og fleksibelt grensesnitt. Denne løsningen vil gjøre det mulig for kundeservicemedarbeidere å trygt oppdatere JSON-filer i Azure DevOps, uten å foreta uautoriserte endringer i koden, og sikre en effektiv håndtering av disse endringene via CI (Continuous Integration)-pipelines.



Figur 2: Løsningen

1.4 Det nye produktet

Produktet som skal utvikles for å løse disse problemene er en webapplikasjon som vil bli brukt av kundeservicen i bedriften. Ved hjelp av denne nye webapplikasjonen vil vanlige og relativt enkle problemer som sendes direkte fra kundeservice til OPS bli løst, og dermed redusere arbeidsmengden til de ansatte i OPS-avdelingen. Dette vil føre til besparelser både i tid og ressurser i bedriften. For eksempel, hvis vi tar for oss en enkel oppgave som å endre påloggingsinformasjon, vil en kundeserviceansatt kunne utføre nødvendige handlinger gjennom webapplikasjonens grensesnitt, og påloggingsinformasjonen til kunden vil bli oppdatert på kundenes database på Azure. På denne måten kan oppgaven enkelt utføres, samtidig som OPS-ansatte får mer tid til å arbeide med mer komplekse problemer. I tillegg er målet med dette

prosjektet å øke kundetilfredsheten ved å tilby raskere og mer effektiv håndtering av kundehenvendelser.

2. Sentrale valg og avgjørelser

Kapittelet setter søkelys på de viktige valgene og beslutningene som ble gjort tidlig i prosjektets løp. Disse tidlige avgjørelsene, som inkluderer avklaring av forventninger, fastsettelse av interne forutsetninger, og valg av teknologi, har spilt en avgjørende rolle i prosjektets utforming og suksess.

2.1 Forventningsavklaring

Under pre-sprint fasen opprettholdt gruppen jevnlig kontakt med produktets eier, og det ble etablert gjensidige forventninger mellom gruppen og produkteieren. Produkteieren, Sikri, formidlet visse forventninger til gruppen, inkludert krav om at gruppen skulle arbeide to dager på kontoret og én dag eksternt. Det ble understreket viktigheten av å gi beskjed i tilfelle gruppen ikke kunne møte, svare på henvendelser innen rimelig tid, være ærlige, overholde regler på kontoret, ivareta taushetsplikt, informere om utfordringer gruppen støtte på, og oppdatere tidsplaner. Disse forventningene ble nøye dokumentert ved hjelp av MoSCoW-metoden og omtalt i *kapittel 4.5* under systemkrav.

På sin side stilte gruppen visse krav til Sikri, inkludert krav om rask respons på spørsmål og meldinger. Det ble også etablert en felles kommunikasjonskanal gjennom Slack. Produkteieren ble forventet å delta fysisk på relevante møter, som sprint review og statusmøter med gruppen og deres veileder fra UiA. Videre forventet gruppen å motta grundig og konstruktiv kritikk fra produkteieren, i håp om at dette ville bidra til at produktet oppfylte deres forventninger.

2.1.1 Forventninger rundt teknologibruk

En sentral beslutning involverte valg av teknologisk plattform. Sikri besatt allerede kompetanse innen C# og Sikris ansatte med ekspertise var tilgjengelige som ressurser for gruppen. Til tross for denne veiledningen ble gruppen tilbudt betydelig handlingsrom for valg av teknologi. I konsensus besluttet gruppen å benytte seg av C# til backend og React til frontend. Dette valget ble motivert av gruppenes sertifiseringer innen React, betydelig erfaring med rammeverket, og en følelse av trygghet knyttet til dets bruk. I tillegg ble det avgjort at gruppen skulle implementere pipeline-teknologi.

Selv om gruppen ikke besitter omfattende erfaring med pipeline-teknologi, blir dette sett som en stimulerende reise preget av læring, gitt at erfarne utviklere fra Sikri vil veilede dem gjennom prosessen.

2.1.2 Kvalitet

I *seksjon 1.2* presenterte vi en oversikt over prosjektet. Prosjektets omfang er betydelig bredt, og gruppen har identifisert mange mulige ideer for implementasjon. Imidlertid har gruppen nøye vurdert kompleksiteten knyttet til å forsøke å realisere alle aspekter av prosjektet innenfor en begrenset tidsramme.

Proof of Concept (PoC) er en produktvisning som tester om en idé kan realiseres, i stedet for å undersøke markedsetterspørselen eller bestemme den beste produksjonsmetoden; den vurderer ideens gjennomførbarhet og finansielle potensial (Gillis, 2023). Produktleder Sikris forventninger til prosjektet inkluderer utviklingen av en fungerende PoC (proof of concept), med intensjonen om å videreutvikle systemet etter vårt bidrag. Av den grunn har gruppen besluttet å implementere en helhetlig systemløsning, men med et fokus på å unngå unødig kompleksitet.

For å sikre at de forventede resultatene møter kravene, ble kvalitetssikring inkludert jevnlig i løpet av sprintprosessen. Kvalitetssikringen omfattet inspeksjon, testing, rapportering av resultater og korrigering av avvik eller endringer for å sikre at prosjektet oppfyller kravene. I tillegg ble det gjennomført flere tester med brukere og eksperter for å sikre kvaliteten på prosjektet.

2.1.3 Kontrakter

Teamet består av studenter som har god erfaring med å samarbeide på andre prosjekter. Basert på denne erfaringen ble gruppemedlemmene raskt enige om en felles gruppekontrakt, som nevnt i *appendiks 9*. Denne kontrakten inkluderer retningslinjer for å håndtere utfordringer som sen levering av arbeid eller manglende overholdelse av regler. Hensikten med kontrakten er ikke å skremme, men å tjene som en referanse for å effektivt håndtere utfordringer uten å belaste hele gruppen. En tilsvarende avtale ble også inngått med Sikri og ansvarlige fra UiA for å sikre gjensidig utbytte, med klart definerte ansvarsroller og kontaktpersoner.

2.2 Prosjektstyring

Gruppen valgte å implementere en agil prosjektstyringsmetode for å gjennomføre prosjektet. Agile-metodologien gir økt fleksibilitet, rask tilpasningsevne og kontinuerlig forbedring i prosjekter, noe som er avgjørende i dagens komplekse og dynamiske forretningsmiljø (Highsmith, 2002)

2.2.1 Scrum

Scrum er et rammeverk for adaptiv problemløsning som involverer en Produkteier som prioriterer oppgaver, et Scrum-team som leverer verdi i sprinter, og iterativ gjennomgang og justering med interessenter (Scrum Guides, 2020). Gruppen gjennomførte daglige møter, ukentlige møter og

arbeidet med sprinter i samsvar med Scrum-reglene. Valget om å benytte Scrum ble gjort for å sikre den mest effektive veien til å levere oppgaven på best mulig måte.

Scrum-verdiene Forpliktelse, Fokus, Åpenhet, Respekt og Mot (Scrum Guides, 2020) ble nøye ivaretatt og hjalp gruppen med å støtte hverandre, opprettholde setter søkelys på oppgaven og nå målene sammen. Respekt ble vist både for hverandre og for produktets eier, samt for brukernes kommentarer og tilbakemeldinger. Gruppen oppmuntret til stadige forbedringer.

2.2.2 Rollefordeling

Teamets samarbeidsforhold, forsterket gjennom flere prosjekter, har demonstrert at en flat hierarkisk struktur, forsterket med tydelig definerte roller, fremmer effektivitet, særlig under perioder med høy arbeidsbelastning.

I den innledende fasen av prosjektet ble det etablert en lederfunksjon for overordnet koordinering. Scrum Master-rollen ble besluttet å rotere blant teammedlemmene for å gi alle en mulighet til å oppleve og forstå prosjektledelse på nært hold. Dette tiltaket sikret at hvert medlem fikk praktisk erfaring med utfordringene og ansvarsområdene tilknyttet rollen som Scrum Master, og bidro til en balansert fordeling av administrative oppgaver.

I tillegg ble det bestemt at Scrum Master skulle tildele spesifikke personer ansvaret for å forberede møtereferater og håndtere kommunikasjon med produkt-eiere og veiledere. Dette systemet sørget ikke bare for vedvarende kommunikasjon og koordinasjon gjennom prosjektet, men hjalp også teammedlemmene med å utvikle sine administrative og ledelsesmessige ferdigheter.

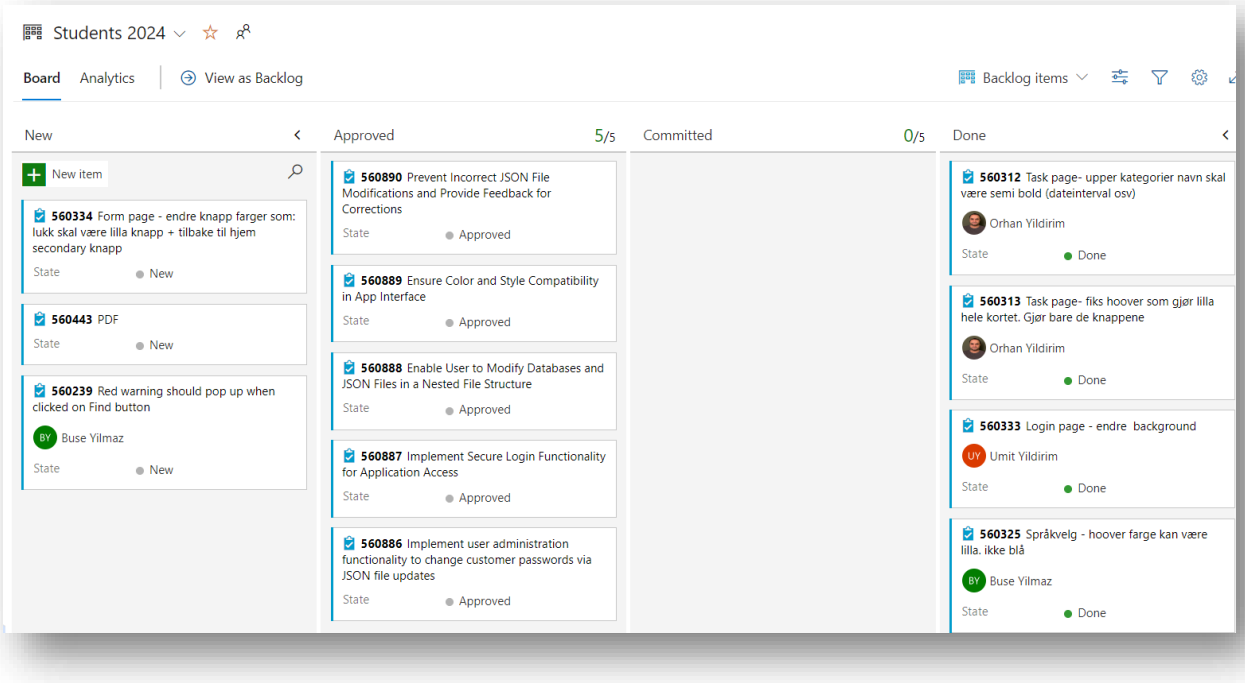
Ved å endre Scrum Master ved hver sprint, fikk teammedlemmene mulighet til å tilegne seg ulike perspektiver og ledelsesteknikker, noe som fremmet en sterk samarbeidskultur og bidro til prosjektets overordnede suksess.

2.2.3 Prosjektstyringsverktøy

Azure DevOps er et omfattende sett med verktøy levert av Microsoft som støtter alle faser av programvareutviklingsprosessen. Det støtter prosjektplanlegging, koding, testing og distribusjon. Azure DevOps tar en helhetlig tilnærming til programvareutvikling og tilbyr sømløs integrasjon med Microsoft-produkter (Microsoft Azure, u.d). Plattformens evne til å enkelt integrere med tredjepartsverktøy gjør den til et fleksibelt og kraftig verktøy for moderne programvareutvikling. Gruppen valgte å bruke Azure DevOps fordi Sikri-ansatte også bruker denne plattformen, noe som påvirket vårt valg. Som sett i figur 3 brukes Sprint-tavler til å visualisere arbeidsflyten og fremgangen av oppgaver. Disse tavlene er organisert i flere kolonner for å reflektere de forskjellige stadiene i arbeidsprosessen.

I prosjektstyringsprosessen gjennomgår teamet oppgavene før hver sprint. Disse oppgavene legges til av teammedlemmene og markeres som 'foreslått'. Scrum Master går deretter gjennom

hver oppgave for å sikre at de er klart definert og i tråd med prosjektets mål før de godkjenner dem for arbeid. Etter godkjenning endres tilstanden til oppgavene til 'godkjent', hvilket betyr at de er klare til å startes på. Underveis i sprinten, når arbeidet på en oppgave påbegynnes, endres tilstanden til 'under arbeid'. Når alle kriteriene for en oppgave er møtt, markeres den som 'ferdig', og oppgaven kan flyttes til den fullførte kolonnen. Dette systemet gjør det mulig for teamet ikke bare å holde styr på fremdriften for hver enkelt oppgave, men også å gi alle interessenter en klar og oppdatert oversikt over prosjektets status.



Figur 3: Sprint board på Azure DevOps

2.2.4 Kommunikasjonsverktøy

Teamet har anerkjent behovet for å etablere regelmessig kommunikasjon med ulike kontakter, spesielt med Sikri og deres veileder, for å styrke prosjektforvaltningen og samarbeidet. Basert på tidligere erfaringer med ulike teknologiske plattformer, har teamet tilpasset sitt kommunikasjonsoppsett for å møte prosjektets varierte behov.

Zoom benyttes for effektiv og målrettet daglig informasjonsutveksling. Gjennom daglige 15-minutters stand-up møter, som avholdes på denne plattformen, sikres det at alle teammedlemmer er oppdaterte på individuelle fremganger, utfordringer identifiseres og neste skritt koordineres effektivt.

Discord støtter kontinuerlig og uformell dialog som fasiliteter agil planlegging. Dette verktøyet hjelper med å holde kommunikasjonslinjene åpne for løpende og spontane diskusjoner, noe som er vitalt for å tilpasse seg raskt til endringer i prosjektets ramme.

Slack fungerer som hovedkanalen for direkte og effektiv kommunikasjon med Sikri. Denne plattformen spiller en kritisk rolle i å opprettholde en jevn strøm av informasjon og sikrer rask respons på henvendelser, noe som er avgjørende for å opprettholde momentum i prosjektet.

For kommunikasjon med universitetsveiledere, benyttes Microsoft Teams og e-post/chat. Dette sammensatte verktøyet muliggjør en hybrid tilnærming til kommunikasjon som kombinerer både formelle og uformelle kommunikasjonsformer, og sikrer at teamet kan ha regelmessig og fleksibel kontakt med veiledere.

Ukentlige ansikt til ansikt møter med Sikris veileder og regelmessige møter i Sikris møterom er kritiske for effektiv prosjektstyring. Disse møtene muliggjør direkte kommunikasjon og samhandling, som er essensiell for en grundig gjennomgang av prosjektets status og for diskusjoner om fremtidige retninger. Ved å fremme en åpen og konstruktiv dialog, legges grunnlaget for et sterkt samarbeid, som er til fordel for prosjektets suksess.

Dette kommunikasjonsoppsettet styrker den kollektive forståelsen og sikrer at alle deltakere er synkroniserte med tanke på prosjektets mål og utfordringer, og bidrar til å opprettholde en kultur av åpenhet og samarbeid.

2.2.5 Estimering

I vår oppgave har vi valgt å gjennomføre historiepoeng som vårt hovedverktøy for estimering av oppgaver i programvareutviklingsprosjekter. Vi benytter Fibonacci som system for tildeling av historiepoeng, et system Sikri også bruker. Fibonacci-sekvensen er en tallrekke hvor hvert tall er summen av de to foregående tallene, starter vanligvis med 0 og 1 (Cohn, 2005). Dette systemet, som følger sekvensen 0, 1, 2, 3, 5, 8, 13, hjelper oss med å vurdere kompleksiteten og arbeidsmengden forbundet med hver oppgave, der høyere tall representerer større kompleksitet og usikkerhet. For å gjøre estimeringsprosessen mer interaktiv og nøyaktig, har vi tatt i bruk pokerplanlegging, en konsensusbasert teknikk. I pokerplanlegging samles alle teammedlemmer, og hver person velger et kort som representerer deres estimat for en oppgave, vanligvis i form av historiepoeng i henhold til Fibonacci-sekvensen.

For eksempel, når teamet skal estimere oppgaven med å legge til en liste med oppgaver på oppgavesiden, vurderer de kompleksiteten involvert i å designe og implementere funksjonaliteten på klient-siden. En foreslår først 3 poeng basert på deres antagelse om oppgavens enkelhet, mens en annen foreslår 5 poeng og peker på utfordringer med å sikre god brukervennlighet og interaktivitet i brukergrensesnittet. Etter en diskusjon, hvor teamet overveier alle tekniske og brukeropplevelse aspekter, enes de om enighet på 5 poeng. Denne metoden fremmer diskusjon og fører til en dypere forståelse av oppgaven, noe som resulterer i mer nøyaktige estimater

(Grenning, 2002). Selv om tradisjonelt mange team har eksperimentert med forskjellige metoder for å tildele historiepoeng, har vår beslutning om å følge Fibonacci-sekvensen vist seg å være effektiv. Den matematiske progresjonen i Fibonacci-sekvensen gir et intuitivt rammeverk for å forstå og kommunisere om oppgavenes relative størrelse og kompleksitet (Cohn, 2005).

Estimering av oppgaver er avgjørende i smidig utvikling. Det hjelper teamet med å forstå og planlegge arbeidsmengden bedre, og sikrer at alle har en felles forståelse av oppgavens omfang. Det er viktig å huske at disse estimatene er nettopp det – estimater. De gir en indikasjon på størrelse og kompleksitet, ikke en eksakt tidsramme (Rubin, 2013).

2.2.6 Risikoanalyse

Risikoanalyse er en teknikk designet for å minske usikkerheten ved å evaluere potensielle risikoer i et prosjekt. Dette bidrar til en dypere innsikt i hvordan man kan håndtere eller unngå disse risikoene hvis de inntreffer (Karlsen, 2018, s. 431).

Gruppen har gjennomført en risikoanalyse som er dokumentert i *Appendiks 3*, hvor den anvendte metoden er en '5x5 Risk Matrix'. Denne metoden illustrerer risikonivået fra lavt til høyt ved å integrere sannsynligheten for og konsekvensene av en hendelse (SafetyCulture, 2023b). Detaljene og spesifikasjonene av denne matrisen kan finnes i *Appendiks 4*. I tillegg har gruppen ført en logg over risikohendelser som har oppstått under prosjektets gang. Disse hendelsene vil bli ytterligere analysert i *kapitel 7.6* om Risikohåndtering.

2.2.7 Sprint-plan/Forventet fremgang

I løpet av prosjektutviklingen implementerte teamet en smidig metodikk, strukturert gjennom en serie av sprints, for å sikre effektiv og målrettet fremdrift. En sprint defineres som en tidsbegrenset periode dedikert til å fullføre et forhåndsdefinert sett med oppgaver, et sentralt element i smidig utvikling. Prosjektet besto av totalt 8 sprints, hver med spesifikke fokusområder fra den innledende prosjektstarten og utforskning av problemområdet, gjennom analyser og designarbeid, til prototyping og brukertesting. Deretter fortsatte prosessen med grunnleggende utvikling, funksjonsforbedringer, omfattende testing, dokumentasjon og sikkerhetsarbeid, og ble avsluttet med forberedelser til den endelige presentasjonen. Denne systematiske tilnærmingen tillot teamet å navigere effektivt gjennom kompleksiteten i programvareutviklingsprosessen, sikre høy kvalitet på leveransene og tilpasse seg endringer og nye innsikter underveis.

Sprint Nummer	Tittel
Pre-sprint	Prosjektstart og utforskning problemdomene
1	Analyse og design
2	Prototyping og brukertesting
3	Grunnleggende utvikling
4	Utvikling av funksjoner
5	Utvikling, Dokumentasjon og Sikkerhet
6	Kvalitetssikring
7	Deployment
8	Fullføring og forberedelse til avsluttende prosjektpresentasjon

Tabell 1: Sprintplanlegging

2.3 Teknologi

I denne delen vil det fokusere på programmeringsspråkene, bibliotekene og andre teknologier som skal brukes i utviklingen, og det vil bli gitt forklaringer på hvordan og hvorfor de brukes i prosjektet.

2.3.1 Frontend og backend

Som tidligere nevnt, vil utviklingen bruke React til frontend og ASP.NET Core til backend. Valget av disse språkene er basert både på språkene som bedriften er vant til å bruke i sine egne utviklingsmiljøer, og på gruppemedlemmenes utdannelse og erfaring med disse språkene. "React and ASP.NET Core"-prosjektmålet som finnes i Visual Studio Code 2022 vil bli brukt som mal for prosjektet.

React er et JavaScript-bibliotek som brukes til å lage brukergrensesnitt basert på en komponentbasert arkitektur. Den komponentbaserte arkitekturen gjør det enkelt å håndtere brukergrensesnittets tilstand og logikk på en modularisert og gjenbrukbar måte. Hver komponent i React kan håndtere sin egen tilstand og logikk, noe som gjør koden mer lesbar og vedlikeholdbar (Abramov & Nabors, 2023).

ASP.NET Core er en tverrplattform, høytytende, åpen kildekode rammeverk for å bygge moderne, sky-aktiverte, internett-tilkoblede applikasjoner (Roth et al., 2023). Det er kjent for sin ytelse og skalerbarhet og støtter en rekke funksjoner som autentisering, sikkerhet, og databasetilkoblinger. ASP.NET Core er også plattformuavhengig, noe som betyr at den kan kjøre på ulike operativsystemer som Windows, Linux og macOS.

2.3.2 Database

I prosjektet vårt valgte vi å ikke bruke en tradisjonell database som MySQL for lagring av data. Grunnen til dette er at målrettede filer og data allerede er lagret i Azure. Dette eliminerer behovet for en separat databaselagring, da Azure tilbyr effektiv tilgang og håndtering av disse filene. Dette forenkler arkitekturen og reduserer kompleksiteten ved å unngå unødvendig datalagring og håndtering.

2.3.3 Utviklingsmiljø

Basert på omfattende diskusjoner internt i teamet og rådgivning fra veiledere, ble det besluttet å bruke Visual Studio 2022 som hovedverktøy for programmeringsarbeidet (IDE - Integrated Development Environment). Dette valget ble tatt fordi prosjektet var basert på en Azure utviklingsplattform, og Visual Studio er kjent for sin sømløse integrasjon med Azure. Azure tilbyr også tilgang til andre nøkkelteknologier som ble tatt i bruk, inkludert Azure DevOps. Azure DevOps fungerte som et sentralt system for sprintplanlegging, der vi kunne spore fremgangen og organisere arbeidsoppgaver og brukerhistorier for de ulike sprintene.

DevOps er en programvareutviklingsmetodikk som fokuserer på samarbeid, kommunikasjon og integrasjon mellom utviklingsteam og IT-operasjonsteam (Božić, 2023). Den er også integrert i Azure. Tidligere var Azure hovedsakelig forbundet med skyinfrastruktur for webhostingpakker, men med Azure DevOps har vi kunnet dra nytte av et rammeverk som fremmer effektivt samarbeid på tvers av fagområder, noe som har vært avgjørende for vår suksess i utviklingsprosessen (Azure, u.å.).

3. Kvalitetssikring

Å bevare høy kvalitet i prosjektet var et sentralt anliggende både for teamet og interessentene. I dette kapittelet er det detaljert hvordan kontinuerlig dialog og samarbeid bidro til å opprettholde disse kvalitetsstandardene gjennom hele prosjektprosessen. Et vesentlig element i denne sammenheng var den regelmessige og åpne kommunikasjonen med bedriften, som sikret at prosjektet stadig møtte deres forventninger og standarder. I tillegg implementerte gruppen flere interne strategier for kvalitetssikring, deriblant en effektiv organisering av teamet og en robust praksis for versjonskontroll.

3.1 Kommunikasjon

Velorganisert kommunikasjon og samarbeid innen kvalitetskontroll og kvalitetssikring er grunnleggende faktorer for å fullføre prosjekter med suksess (Larson & Gray, 2021). I utviklingsprosessen av prosjektet, utgjør effektiv kommunikasjon med veileder/produkteier en fundamental byggestein for prosjektets suksess. Daglige meldinger og kommunikasjon gjennom Slack og Discord har bidratt til en dypere forståelse av prosjektet, korrekt oppfattelse av

forventningene og forebygging av potensielle feil i en tidlig fase. Internt i gruppen er kommunikasjon også av vesentlig betydning for prosjektets fremdrift. Daglige møter ved dagens slutt legger til rette for deling av informasjon om dagens arbeid og planlegging for den kommende dagen. Disse møtene fremmer en rask og effektiv fremgang, opprettholdelse av forventet kvalitet og sikrer jevn progresjon gjennom hele prosessen.

I denne sammenhengen har utviklingsteamet hos Sikri AS nøye tilrettelagt for effektiv kommunikasjon med både interne og eksterne interessenter, for å sikre riktig retning og fremgang i prosjektet. Effektiv kommunikasjon garanterer ikke bare at prosjektet fullføres innen tidsfristen og opprettholder forventede kvalitetsstandarder, men sikrer også at alle teammedlemmer har full forståelse av prosjektets mål og prosesser.

3.2 Kodestandard

For å sikre et høyt kvalitetsnivå på produktet, tilpasset gruppen seg Sikris detaljerte kodestandarder for C#. Dette initiativet gjorde det mulig for oss å skrive kode som lett kunne vedlikeholdes og videreutvikles av Sikris egne utviklere, samtidig som det sikret at koden var i tråd med bransjestandarder. Implementeringen av disse standardene, som inkluderte bruk av 4 mellomrom for innrykk (ikke indentasjon) og preferanse for å unngå ekstra linjer ved filslutt, bidro til en ryddig og godt organisert kodebase.

Disse tilpasningene understreket viktigheten av konsistens og lesbarhet i koden, noe som ikke bare forbedrer kvaliteten, men også fasiliteter mer effektivt samarbeid og vedlikehold. Videre tillot adopteringen av bestemte preferanser, slik som forsiktig bruk av 'var'-nøkkelordet og expression-bodied medlemmer (ikke funksjoner) for enklere metoder, gruppen å skrive moderne og vedlikeholdsvennlighet kode (Salian, 2020).

Ved å nøye følge disse praksisene fra starten, sikret gruppen en sømløs integrasjon med Sikris eksisterende kodebase, og etablerte et solid fundament for fremtidig utvikling og samarbeid.

3.3 Versjonskontroll

Versjonskontrollsystemer spiller en nøkkelrolle i å holde oversikt over endringer i koden og forebygger tidlig konflikter som kan forårsake skade eller forstyrrelser (Microsoft, 2022). I løpet av dette prosjektet, basert på diskusjoner mellom teamet og eieren av prosjektet, ble det bestemt å benytte Git i Azure Repos, en praksis også omfavnet av Sikri AS, for å koordinere koden. Det ble besluttet at hver 'commit' skulle systematisk kobles til et oppgavenummer og en oppgavebeskrivelse i Azure Board. Denne tilnærmingen gjør det lettere å identifisere hvilke handlinger og eventuelle utfordringer som kan oppstå fra kodefusjoner ved å peke ut spesifikke 'commits' eller oppgaver. Dersom koden blir korrupt på grunn av en feilaktig 'merge', gjør dette

systemet det mulig å raskt finne og rette feilen, og dermed sikre en jevn fremgang i prosjektet. Dessuten har teamet implementert en policy for grenhåndtering som forutsetter at sammenslåinger til hovedgrenen først må gjennomgås og godkjennes av Scrum masteren eller et annet medlem av teamet.

3.4 Team-oppsett

Prosjektet omfattet alle faser i en programvareutviklingssyklus, inkludert analyse og ressursplanlegging, design og prototyping, utvikling, testing og deployment. Derfor var tid og ressursstyring av stor betydning. Ved prosjektets start arbeidet hele teamet tett sammen under analyse- og designfasene for å sikre en felles forståelse og visjon for prosjektet. Imidlertid, under programvareutviklings- og testfasene, ble det besluttet at det ville være mer effektivt å dele teamet inn i en frontend- og en backend-gruppe for å bedre håndtere tid og ressurser. Dette tiltaket ikke bare akselererte arbeidsprosessen, men også forbedret kvaliteten på arbeidet som ble utført.

Gruppemedlemmenes ferdigheter og erfaringer ble tatt i betraktning ved tildeling av oppgaver. Noen medlemmer tok på seg frontend-utvikling på grunn av deres tidligere erfaring med JavaScript-biblioteker som React. Andre medlemmer, med erfaring innen backend og testing, tok ansvar for disse områdene. Ettersom prosjektet både var et reelt prosjekt og en læringsmulighet, oppstod det et behov for kunnskapsdeling blant gruppen. For å møte dette behovet holdt teamet ukentlige evalueringsmøter hvor de delte informasjon om hva de hadde jobbet med den uken, inkludert bruk av kode, funksjoner og biblioteker. Disse møtene sikret at alle teammedlemmer hadde oppdatert kunnskap om hvert aspekt av prosjektet.

I tillegg hadde teamet daglige møter som varte i maksimalt 15 minutter for å dele oppdateringer om hva de hadde jobbet med, fremtidige planer, og eventuelle problemer de møtte på. Disse daglige møtene styrket kommunikasjonen innad i teamet og økte motivasjonen.

Gruppen valgte å dele scrum master-rollen mellom hvert medlem for å oppnå en flat struktur i gruppen. Dette valget ble motivert av tanken om at mennesker blir mer engasjert og motivert når de har større ansvar for sine handlinger. Forskning antyder at dette kan føre til redusert fravær og lavere turnover, som i sin tur kan føre til positiv utvikling i hele organisasjonen (Hoel, 2006). Denne tilnærmingen hjalp oss med å bedre forstå scrum master-rollen og fremme bedre samarbeid. Alle gruppemedlemmene delte fritt sine meninger og var en integrert del av beslutningsprosessen. Dette skapte en følelse av økt motivasjon og dedikasjon til prosjektet.

3.5 Testing

Testing er en systematisk aktivitet rettet mot å validere og verifisere at et programvaresystem møter de definerte kravene. Denne prosessen er avgjørende for å identifisere feil og mangler før

produktet når sluttbrukerne, og spiller derfor en sentral rolle i kvalitetssikringen av programvare (Lewis, 2019). Kvalitetssikring består av prosedyrer og praksiser som sikrer at et produkt eller en tjeneste vil tilfredsstille de forventede kravene. Testing er direkte knyttet til kvalitetssikring da det hjelper til med å opprettholde produktets integritet og sikrer at alle funksjoner virker som de skal under forskjellige betingelser (Patton, 2020).

I vårt prosjekt førte en dedikert tilnærming til testing til betydelig bedre kvalitet. Vi utførte brukertesting for å sikre at programvaren var intuitiv og brukervennlig, og eksperttesting for å verifisere at systemet oppfylte tekniske og sikkerhetsmessige standarder. Disse testmetodene ga verdifulle innsikter som førte til forbedringer i programvarens design og funksjonalitet. Gjennom en serie med enhetstester verifiserte vi individuelle komponenter, og med integrasjonstester sjekket vi komponentinteraksjonene. Systemtester ble også benyttet for å evaluere systemets oppførsel mot de fullstendige kravene (Ahmed & Kaur, 2022).

Denne systematiske tilnærmingen til testing sikret at programvaren ikke bare møtte de tekniske spesifikasjonene, men også brukernes forventninger til ytelse og pålitelighet, som er avgjørende for et vellykket programvareprodukt. Mer detaljert informasjon om vår tilnærming testene vil bli diskutert i en senere del av rapporten under tittelen testing.

3.6 Dokumentasjon

Dokumentasjon inneholder informasjon og publikasjoner som hjelper utviklere med å forstå planlegging, koding og implementering av et produkt. Dokumentasjon gjør det mulig for utviklere å forstå, oppdatere og tilpasse programvaren fra grunnen av. Teknisk dokumentasjon detaljerer programvarens arkitektur, brukte teknologier, kodingsstrukturer, API-referanser og integrasjonspunkter (Javapoint, u.å). Dokumentasjon sikrer kvalitet ved å gi en klar og detaljert beskrivelse av programvarens funksjoner og strukturer, som hjelper utviklere med å unngå feil og følge beste praksis. Dette bidrar til å opprettholde programvarens integritet og pålitelighet gjennom hele utviklingsprosessen.

Det er utarbeidet en detaljert kode-dokumentasjon som dekker alt fra installasjon av applikasjonen, avhengigheter, backend-strukturen til legging av nye oppgaver. Denne dokumentasjonen guider utviklere i å forstå hvordan applikasjonen fungerer og hvordan den kan videreutvikles. Dokumentasjonen har bidratt til et bedre utviklingsmiljø for applikasjonen. Det kan finnes i *Appendiks 13*.

4. Analyse og applikasjonsdesign

Denne delen dekker forarbeidet som ble utført før gruppen startet på hovedprosjektet. I analysen bør designeren sette søkelys på designets funksjonalitet, effektene av de brukte metaforene, og hvordan folk vil oppfatte dem (Benyon, 2019, s. 219).

4.1 Forstå Problemdomenet

Det var avgjørende for gruppen å grundig forstå problemområdet applikasjonen skulle adressere før de startet med designprosessen. Uten en klar forståelse av problemet, er sjansen for at den utviklede løsningen faktisk løser de forventede problemene, lav. Derfor, for å oppnå en klar forståelse av brukernes forventninger og behov, gjennomførte gruppen intervjuer med prosjekt eier og noen ansatte i kundeservice hos Sikri AS.

4.2 Intervju

I den første fasen av prosjektet, ble det avholdt flere møter med produktets eier for å oppnå en dypere forståelse av problemområdet. Målet med disse møtene var å få innsikt i produktetseiers behov, forventninger og forretningsmål.

Produkteieren uttrykte ønske om å utvikle en applikasjon basert på prinsippene om brukersentret (user-centered design) og kunne brukes enkelt av brukere uten teknisk kunnskap, og som minimaliserte risikoen for feilbruk. User-centered design er et bredt begrep som beskriver designprosesser hvor sluttbrukernes behov og preferanser tas i betraktning gjennom hele utviklingsprosessen (Abras et al., u.å.). Ifølge Norman, er designerens rolle å lette brukerens opplevelse og sikre at produktet kan brukes intuitivt og effektivt med minimal opplæring (Abras et al., u.å.). Som gruppe 15, arbeidet vi aktivt med produktets eier for å identifisere mulige utfordringer og utforske mulige løsninger med brukersentret design som kunne imøtekomme deres behov.

For å oppnå dette målet, gjennomførte gruppen tre semistrukturerte intervjuer med målgruppen i løpet av sprint 1. Intervjudokumentet finnes under *appendiks 11.8.* Formålet med intervjuene var å få innsikt i brukernes perspektiver, behov og forventninger til applikasjonen. Intervjuene startet med generelle spørsmål om brukernes generelle oppfatning av applikasjonen, før de gikk mer i dybden på spesifikke funksjoner. I tillegg ble kandidatene spurt om forslag til nye funksjoner de ønsket å se i applikasjonen. Disse tilbakemeldingene ble vurdert av gruppen med tanke på implementering i utviklingsprosessen. Intervjuenes funn og sammendrag ble inkludert som appendiks i rapporten og bidro til å danne et mer helhetlig bilde av målgruppen.

4.3 Persona

Basert på funnene fra intervjuene, utarbeidet gruppen en persona, en fiktiv karakter som representerer potensielle brukere av applikasjonen. Personas er et verktøy som bidrar til å forenkle designoppgaven ved å styre idéprosessen og hjelpe med å oppnå målet om å skape en optimal brukeropplevelse for målgruppen (Dam & Teo, 2024). Figur 4 viser en av personene som gruppen brukte for å få innsikt i den generelle gjennomsnittsbrukeren av applikasjonen.

Informasjonen som ble brukt til å utforme personen ble hentet fra intervjuer og observasjoner gjort i starten av problemområdet perioden.

Figur 4: Persona

4.4 Brukerhistorier

Brukerhistorier benyttes for å styrke kommunikasjon mellom utviklere og sluttbrukere og kan hjelpe til med å konkretisere funksjonaliteter for et system under utvikling (Dimitrijević, Jovanović, & Devedžić, 2015). Gruppen bruker disse brukerhistoriene for å prioritere oppgaver basert på verdien de tilfører applikasjonen og for å optimalisere tid i det agile rammeverket. Ved å følge MoSCoW-metoden sikrer de at kravene tilfredsstiller kundenes behov innenfor gitte tidsrammer (Hudda, Mahajan, & Chopra, 2016).

Brukerhistoriene i dette prosjektet er basert på intervjusvar fra brukere og møter med produkteieren. De er alle strukturert etter samme format: "Som X, ønsker jeg Y, for å oppnå Z". Målet er å fokusere på brukerens og produkteierens behov.

Tabell 2 gir en oversikt over utvalgte brukerhistorier i dette prosjektet. En fullstendig tabell finnes i *appendiks 6*.

ID	Prioritering	Brukerhistorie	Argument
1	Must Have	Som en bruker ønsker jeg å kunne oppdatere innstillinger for en kunde gjennom json-filer.	Denne funksjonen er prioritert som «Must Have» fordi dette vår siste målet av hele prosjektet. Dette er essensielt for effektiv håndtering av saker og kundeinformasjon.
2	Must Have	Som en bruker ønsker jeg en brukervennlig, minimalist grensesnitt og enkel navigasjon for å forbedre min generelle opplevelse med applikasjonen.	Denne funksjonen rangeres som et «Must Have» fordi den forbedrer applikasjonsopplevelsen, selv for brukere uten teknisk kunnskap.
4	Must Have	Som en bruker ønsker jeg å kunne logge meg inn i applikasjonen for å sikre at bare autoriserte brukere har tilgang til systemet.	Funksjonen er et Must Have, fordi dette systemet kun skal kunne benyttes av kundesenter hos Sikri.
5	Must Have	Som en bruker, ønsker jeg å kunne oppdatere innstillingene for en sak uten risiko for feilvalidering, for å sikre nøyaktig håndtering av sensitiv og kritisk data.	Denne funksjonen er et Must Have, fordi, dette systemet inneholder sensitiv og kritisk data som ikke må endres feilaktig.
6	Must Have	Som en bruker, ønsker jeg at systemet viser detaljer om endringer på GitHub, for å oppnå bedre sporbarhet og forståelse av implementerte endringer.	Når endringer utføres i taskene i kundesenteret og pushes til GitHub, er det essensielt at commit-loggene inneholder detaljert informasjon om hvem som utførte endringen, når den ble utført, og hva endringen innebar.

Tabell 2: Brukerhistorier

4.5 Systemkrav

Prosjektteamet vårt har gjennomført en grundig analyse av brukerbehov og systemfunksjoner og har klassifisert applikasjonens krav i henhold til MoSCoW-metodologien. Denne metodologien vurderer systemkrav i fire hovedkategorier: 'Must have', 'Should have', 'Could have' og 'Won't have'. Denne klassifiseringen spiller en kritisk rolle i beslutningene rundt prosjektets omfang, tidsplanlegging og ressursfordeling, og er designet for å veilede alle beslutningsprosesser (Hansen & Olsen, 2021).

Krav ID	Prioritering	Krav	Brukerhistorie
1	Must Have	Systemet må tillate oppdatering av kundeinnstillinger via JSON-filer.	Som en bruker ønsker jeg å kunne oppdatere innstillinger for en kunde gjennom json-filer.
2	Must Have	Systemet skal tilby et brukervennlig grensesnitt med intuitivt layout og enkel navigasjon.	Som en bruker ønsker jeg en brukervennlig, minimalist grensesnitt og enkel navigasjon for å forbedre min generelle opplevelse med applikasjonen.
3	Must Have	Systemet må organisere innstillinger i kategorier for å forenkle bruk og spare tid.	Som en bruker, ønsker jeg å kunne finne innstillinger i kategorier, for å forenkle prosessen og bidra til tidsbesparelser.
4	Must Have	Systemet må tilby en sikker innloggings- og autorisasjonsprosess.	Som en bruker ønsker jeg å kunne logge meg inn i applikasjonen for å sikre at bare autoriserte brukere har tilgang til systemet.
5	Must Have	Systemet må tillate oppdatering av innstillinger uten å risikere feilvalidering.	Som en bruker, ønsker jeg å kunne oppdatere innstillingene for en sak uten risiko for feilvalidering, for å sikre nøyaktig håndtering av sensitiv og kritisk data.
6	Must Have	Systemet må vise detaljer om endringer utført i taskene i kundesenteret i commit-loggene på GitHub.	Som en bruker, ønsker jeg at systemet viser detaljer om endringer på GitHub, for å oppnå bedre sporbarhet og forståelse av implementerte endringer.
7	Should Have	Systemet bør tilby tilpasning av grensesnittets farger og temaer.	Som en bruker ønsker jeg å kunne tilpasse grensesnittets farger og temaer, for å oppnå en mer personlig og tilpasset opplevelse.
8	Should Have	Systemet bør tilby flere språkalternativer.	Som en bruker ønsker jeg å kunne velge språk i systemet for å tilpasse opplevelsen til mine preferanser og behov.

9	Should Have	Systemet bør tillate visning av historikk over siste endringer.	Som en bruker, ønsker jeg å kunne se en historikk over de siste endringene, for å ha tilgang til relevant informasjon om systemets endringer.
10	Could Have	Systemet bør støtte samtidig bruk av QuickFix og Zendesk(Kundeservice Applikasjon).	Som en bruker, ønsker jeg å bruke QuickFix og Zendesk samtidig, for å forbedre brukeropplevelsen, selv om dette ligger utenfor prosjektets hovedomfang.
11	Won't Have	Systemet skal ikke tillate innlogging fra flere enheter samtidig.	Som en bruker, ønsker jeg begrensninger for innlogging fra flere enheter samtidig, for å oppnå at applikasjonens mål ikke inkluderer mobilbruk.

Tabell 3: Brukerhistorie med MoSCoW-hierarki

4.6 Skisser

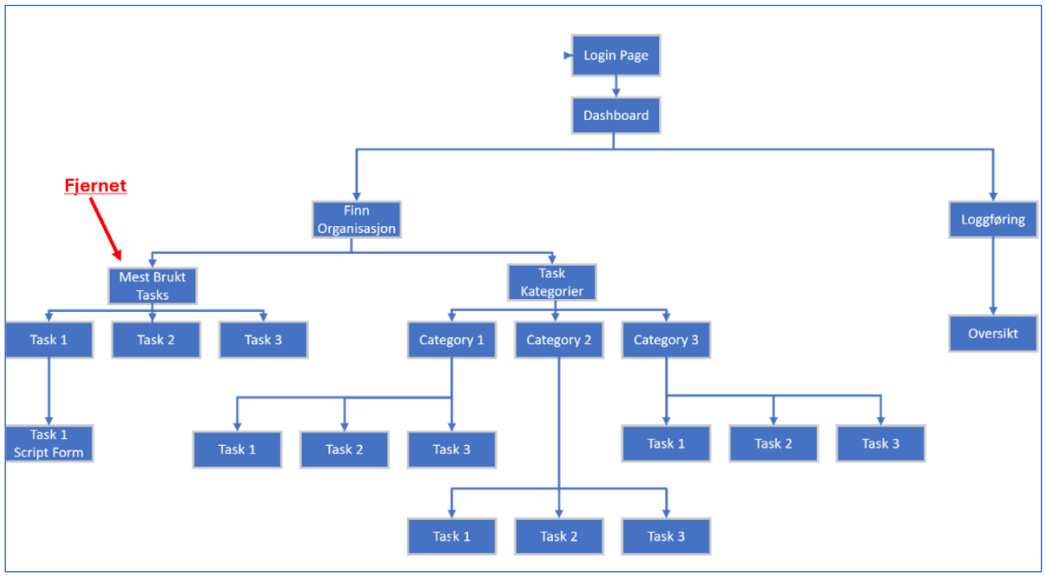
Skissering er en grunnleggende teknikk som bringer abstrakte ideer til liv, og avslører svakheter og utfordringer ved å visualisere dem på papir (Benyon, 2019, s.184). I tillegg til å avdekke mulige utfordringer, fungerer skissering som en kraftfull katalysator for kreativ tenkning og innovasjon. Ved å overføre tanker og konsepter til fysiske bilder, kan designere se ideene sine fra nye perspektiver. Ofte kan de oppdage løsninger og tilnærminger som tidligere ikke var tenkt på (Benyon, 2019, s.184).

I løpet av sprint 1 forsto vi problemet gjennom intervjuer med kunder og ideutvekslinger innad i teamet. Forståelsen av problemet ga alle forskjellige ideer for løsninger, og vi ba derfor alle om å forberede skisser for å kunne vurdere disse ideene. Vi evaluerte de forberedte skissene i møtene våre i sprint 1 og prøvde å finne de mest passende og beste løsningene. Denne fasen ga oss ikke bare nye ideer, men hjalp oss også med å se potensielle problemer i de eksisterende visualiseringene. Det kan finnes i *appendiks 7* for å se skissene som ble laget av teamet.

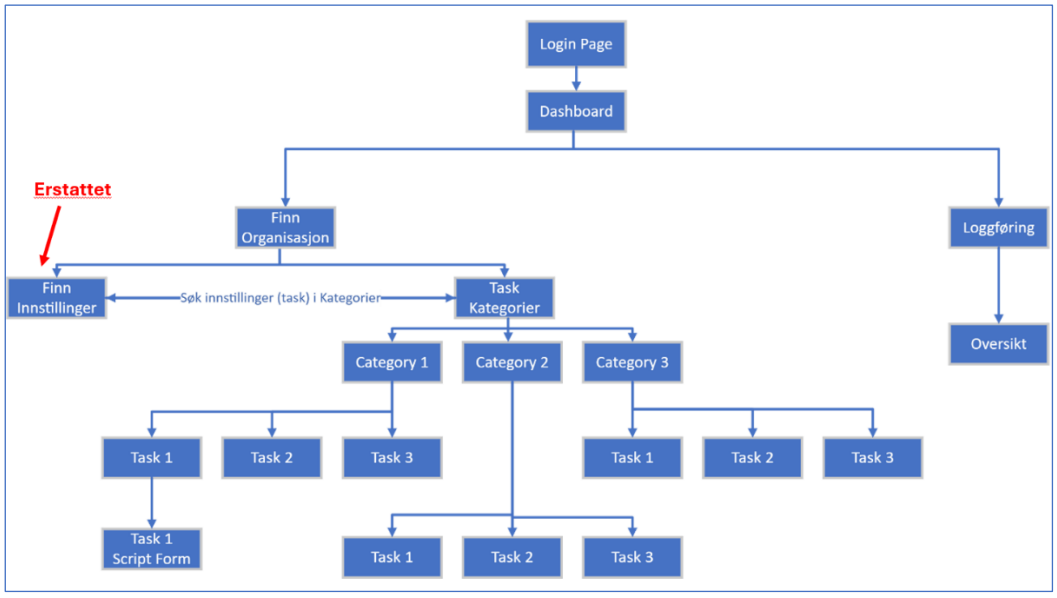
4.7 Navigasjonskart

Navigasjonskartet er et strukturelt skjema som fokuserer på brukeropplevelsen ved å vise flyten og tilgangen mellom sider i et nettsted eller applikasjon (Benyon, 2019, s. 191). Gruppen opprettet et navigasjonskart for å forbedre forståelsen av prosjektet og få en bedre innsikt i hvordan prosessen fungerer. Ved hjelp av navigasjonskartet forsto alle gruppemedlemmene

hvordan systemet fungerer bedre og oppnådde en felles forståelse av prosjektet. Gjennom brukertester og fremdriften i prosessen ble det gjort noen endringer i prosjektet. Figur 5 viser den første versjonen av navigasjonskartet, mens Figur 6 viser den oppdaterte siste versjonen. Etter prosjekteierens forespørsel ble delen "Mest Brukte Tasks" fjernet og erstattet med en søkemodul som lar brukerne søke innen "Kategorier".



Figur 5: Navigasjonskart versjon 1



Figur 6: Navigasjonkart versjon 2

4.8 Designprinsipper

Prosjektet Sikri, betegnet "QuixFix", innebar en nøye vurdering av brukergrensesnittet under designprosessen, hvor det ble tatt hensyn til bransjestandarder og etablerte prinsipper. I utformingen av designstrategien ble innsikt hentet fra David Benyons bok "Designing User Experience" for å etablere grunnleggende prinsipper som forbedrer brukeropplevelsen (Benyon, 2019, s. 113-120). Implementeringen av disse prinsippene er avgjørende for å sikre applikasjonens brukervennlighet, effektivitet og generelle brukeropplevelse.

I vårt prosjekt var de Benyon-prinsippene som var mest fremtredende, "Constraints" (begrensninger) og "Feedback" (tilbakemelding). Disse prinsippene var kritiske ettersom de hjalp til med å forhindre at brukerne gjorde feil i utgangspunktet. Ved å implementere begrensninger kunne vi styre brukerinteraksjonene slik at de unngikk feil handlinger. Til tross for disse forholdsreglene, hvis en feil likevel oppstod, var det essensielt å gi umiddelbar og forståelig tilbakemelding, slik at brukeren raskt kunne rette opp feilen. Dette fokuset på å forhindre og korrigere feil var avgjørende for suksessen til vårt prosjekt.

Målet var å skape et brukervennlig og tilgjengelig grensesnitt ved å minimere kompleksiteten og gjøre nødvendig informasjon lett tilgjengelig. Designet ble nøye vurdert for å redusere sannsynligheten for brukerfeil, ved integrering av kjente elementer som knapper og kortstrukturer, noe som fremmer interaktivitetsletthet. Det ble sørget for at brukerne mottok tydelige og forståelige tilbakemeldinger etter hver handling som ble utført.

For å forhindre brukerfeil, ble det inkludert forklarende merknader i datainngangsfeltene som angir hvilken type informasjon som skal inntastes. Denne tilnærmingen gjorde det mulig for brukerne å forstå hvilken data som kunne legges inn på hvert steg, noe som økte læringsevnen. Ved å analysere tabellen vist i tabell 4 hvor designprinsippene ble prioritert, er det tydelig hvordan hvert prinsipp har formet de forskjellige aspektene av applikasjonen.

Denne gjennomtenkte tilnærmingen reflekterer målet om å forstå og svare på brukernes behov på den mest effektive måten gjennom utviklingsprosessen av QuixFix. Hver designbeslutning ble tatt med omhyggelig hensyn til disse grunnleggende prinsippene, med mål om å maksimere brukeropplevelsen.

Kriterier	Veldig Viktig	Viktig	Mindre Viktig
Brukervennlighet	X		
Effektivitet	X		
Enkel å forflytte		X	
Enkel å teste		X	

Enkel å vedlikeholde		X	
Fleksibilitet			X
Forståelig		X	
Pålitelighet	X		
Sikkerhet	X		

Tabell 4: Designprinsipper

4.9 Prototype

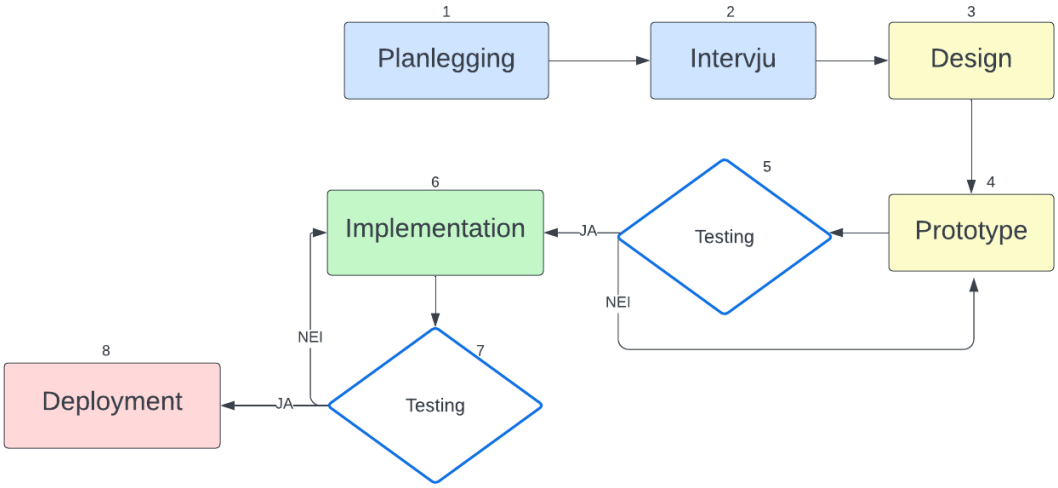
En prototype representerer en konkret, men delvis gjengivelse eller implementering av en systemdesign (Benyon, 2019, s. 195). Prototyping utgjør en kjernekomponent i designprosessen, da den muliggjør rask testing og evaluering av konsepter. Ved å konstruere prototyper, kan designere eksperimentere med forskjellige løsninger og raskt identifisere effektive funksjoner samt områder som krever forbedring.

Under forberedelsene av prototypen for QuickFix, ble Figma anvendt. Applikasjonen, designet som en webapp for bruk på bærbare og stasjonære datamaskiner, ble utformet ved hjelp av Figmas Desktop (1440-1024) modell. Prototypen inneholder totalt åtte hovedsider, inkludert en oppstartsside. Prototypelenken i Figma finnes i *appendiks 12*.

Prototypingsfasen var opprinnelig planlagt til å pågå gjennom sprint 3, men på grunn av en raskere enn forventet skisseringsfase, startet arbeidet med prototypen i den andre halvdelen av sprint 2, og fortsatte med brukertesting frem til slutten av sprint 3. Prototypen ble testet med to brukere, som generelt uttrykte at applikasjonen var enkel og forståelig. Imidlertid bemerket de at noen av termene ikke var de vanligvis bruker eller kjenner til i daglig bruk. Som respons på denne tilbakemeldingen, ble disse termene erstattet med mer kjente ord.

5. Implementering

I gjennomføringsfasen av Quick-Fix prosjektet ble nøye utvalgte teknologier og metoder benyttet for å omsette teoretisk planlegging til praktiske applikasjoner. I tråd med prinsipper om fleksibilitet og robusthet utviklet vi en systemarkitektur designet for å tilpasse seg fremtidige behov og muliggjøre langsiktig vedlikehold, samtidig som vi sikret en problemfri og sikker brukeropplevelse.



Figur 7: Livssyklus for implementering av prosjektet

I figur 7 ovenfor vi illustrerte prosjektets prosess. Prosessen begynte med planlegging og gikk deretter over til intervjudelen for å forstå problemområdet. Etter separate intervjuer med prosjekteier og brukere, produserte vi en løsning som møtte behovene deres. Denne løsningen ble ytterligere konkretisert under designfasen. Ved bruk av Figma i prototypfasen gjorde vi løsningen testbar. Etter å ha testet prototypen, gjorde vi endringer basert på tilbakemeldingene vi fikk, før vi gikk videre til implementeringsfasen. Etter at prototypen hadde bestått siste tester fra eksperter og brukere, ble prosjektet levert for deployment.

5.1 React som Frontend-Rammeverk

I utviklingen av Quick Fix-prosjektet, ble React valgt som hovedrammeverk for å bygge brukergrensesnittet (UI). Dette valget ble basert på React evne til å levere en deklarativ og effektiv brukeropplevelse, og dens popularitet blant moderne webutviklingsverktøy (React, u.å.). React komponentbaserte arkitektur støtter en modulær kodebase, som er en fordel gitt teamets eksisterende erfaringer med biblioteket, og fremmer raskere utviklingsprosesser.

For å gi et responsivt og estetisk tiltalende UI, ble React Bootstrap integrert. Dette biblioteket forener React komponentmentalitet med Bootstrap, et etablert CSS-rammeverk, og tilbyr et bredt utvalg av forhåndsstilte komponenter som knapper og søkefelt (React Bootstrap, u.å.). Material-UI ble også implementert, spesielt for sin robuste samling av komponenter og ikoner, noe som bidrar til konsistens i design og forbedrer brukervennligheten (MUI, u.å.).

Brukerfeedback er en integrert del av brukeropplevelsen, og for å forbedre dette aspektet ble React Toastify valgt for sin effektivitet i å levere klare og intuitive varsler (React Toastify, u.å.). Ved å implementere disse teknologiene, siktes det mot å oppnå en sømløs integrasjon med Sikris

teknologiske økosystem, og legge et solid fundament for fremtidig vedlikehold og skalerbarhet av Quick Fix-applikasjonen.

5.2 Asp .Net

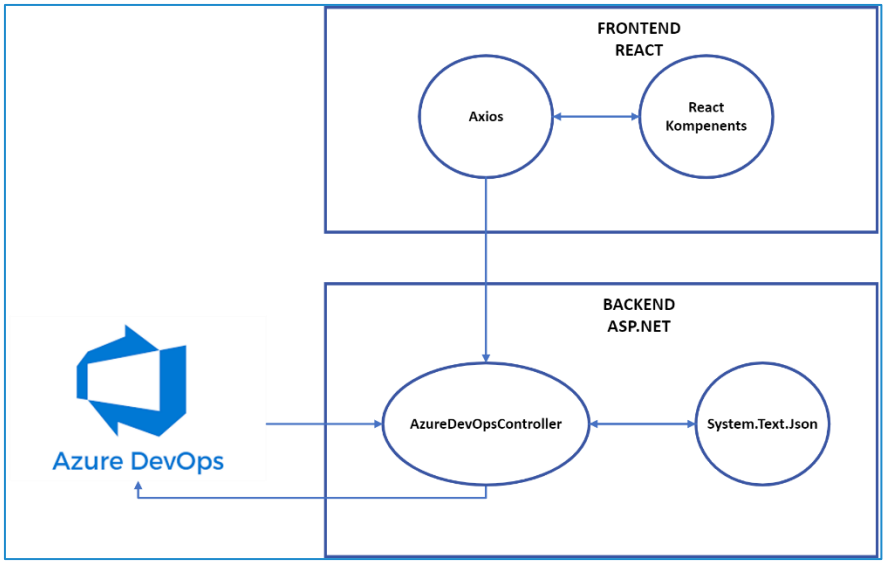
I utviklingen av applikasjonen har ASP.NET-plattformen spilt en nøkkelrolle. ASP.NET er valgt for sin robuste støtte for webapplikasjoner, inkludert viktige aspekter som sikkerhet, ytelse og enkel integrering med andre Microsoft-tjenester som Azure DevOps. Selv om ASP.NET 7 ikke var den nyeste versjonen da vi startet, valgte vi denne teknologien fordi den møtte våre behov på det tidspunktet. Denne beslutningen ble kommunisert til produktansvarlige, og det ble avtalt at en oppgradering til en nyere versjon kan implementeres i fremtiden ved behov. Gjennom bruk av [ApiController]-attributtet, har vi utnyttet konvensjonsbasert routing og binding, som forenkler opprettelsen av RESTful APIer ved å automatisk tilordne HTTP-anrop til controller-handlere basert på metodens signaturer.

For å håndtere avhengigheter effektivt og sikre skalerbarheten til applikasjonen, ble det brukt et IHttpClientFactory for å opprette HttpClient-instanser. Dette mønsteret hjelper til med å unngå vanlige problemer forbundet med livssyklusstyringen av HttpClient-objekter, som utmattelse av systemressurser. Vi implementerte autentisering ved å bruke en personlig tilgangsnøkkel (PAT) for sikker tilgang til Azure DevOps-tjenester, en viktig sikkerhetsmekanisme som beskytter applikasjonen mot uautorisert tilgang. ASP.NET's konfigurasjonssystem ble også brukt for å hente innstillinger som PAT og Azure DevOps-innstillinger på en sikker måte, som støtter både utviklings- og produksjonsmiljøer uten behov for kodeendringer. Applikasjonens controller, AzureDevOpsController, utnytter disse ASP.NET-funksjonene for å tilby et effektivt, sikret API som håndterer konfigurasjonsoppdateringer. Ved å bruke async/await-mønsteret for asynkrone kall til Azure DevOps-tjenester, har vi sikret at APIet vårt håndterer I/O-operasjoner uten å blokkere tråder, noe som forbedrer applikasjonens samlede ytelse og responsivitet.

Sammendragvis representerer valget av ASP.NET for utviklingen av applikasjonen en strategisk beslutning som utnytter plattformens styrker i å bygge sikre, skalerbare og effektive webapplikasjoner.

5.3 Systemarkitektur

Projektet utnytter en lagdelt arkitektur som deler funksjonaliteten mellom frontend og backend, og gjør bruk av moderne teknologier og rammeverk for å levere en robust og skalerbar webapplikasjon. Arkitekturen er designet for å støtte høy ytelse, sikkerhet, og brukervennlighet, samtidig som den opprettholder en klar separasjon mellom brukergrensesnittet og forretningslogikken.



Figur 8: Systemarkitektur for QuickFix

Frontend-delen av applikasjonen er utviklet med React, et moderne JavaScript-bibliotek for å bygge brukergrensesnitt. React gjør det mulig for utviklere å lage komplekse brukergrensesnitt med gjenbrukbare komponenter og state management, noe som fører til en mer organisert og vedlikeholdbar kodebase (React team, u.d.). For å forbedre brukeropplevelsen ytterligere, er frontend integrert med Axios for effektiv datahåndtering og kommunikasjon med backend. React-komponentene er designet for å være responsive og tilby en sømløs interaksjon på tvers av ulike enheter og skjermstørrelser.

På server-siden benytter applikasjonen ASP.NET Core, en kraftig og fleksibel plattform for å bygge webapplikasjoner. ASP.NET Core er valgt for sin ytelse, sikkerhet og evnen til å håndtere høye laster, noe som er essensielt for driftssikkerhet og skalerbarhet. Backend er ansvarlig for å håndtere API-kall, datahåndtering, og integrasjoner med eksterne systemer som Azure DevOps. System.Text.Json brukes for effektiv serialisering og deserialisering av JSON data, som er sentralt for kommunikasjonen mellom frontend og backend.

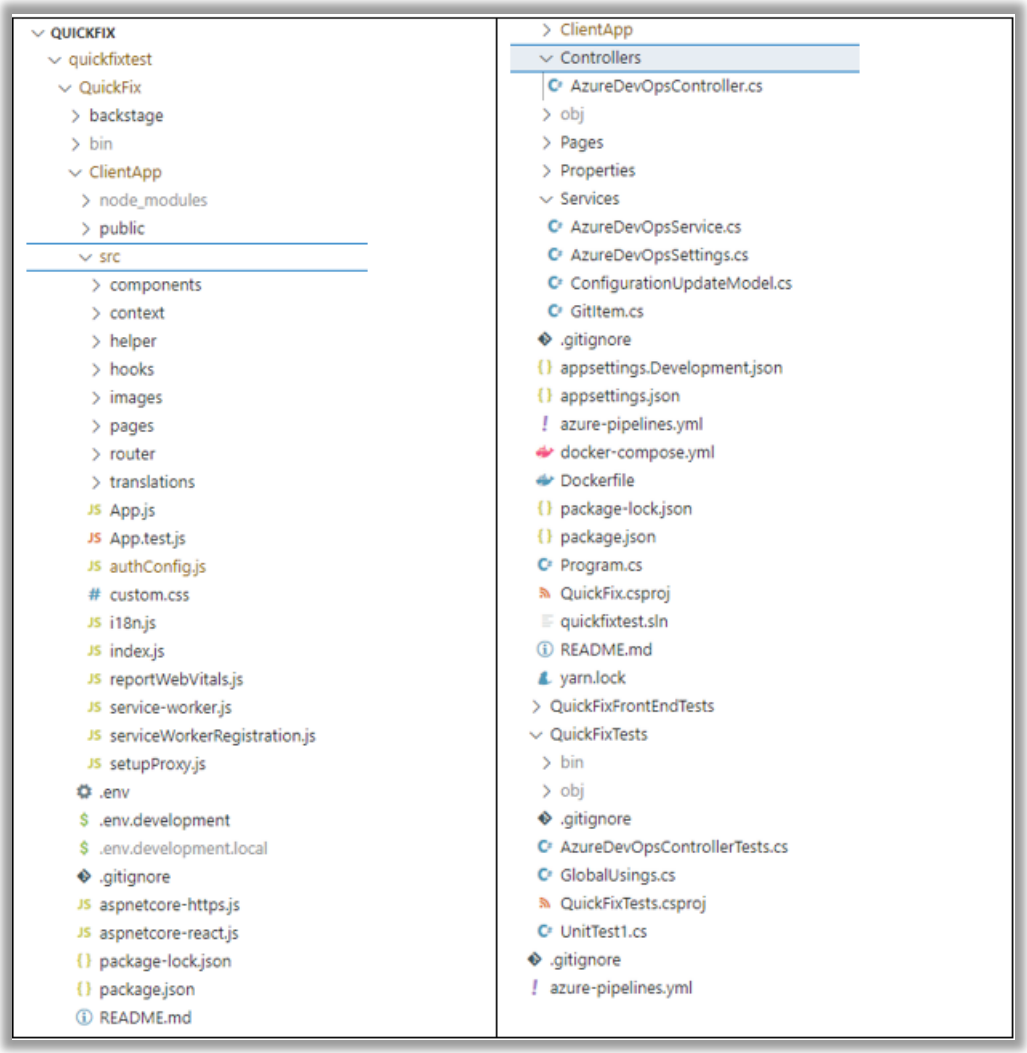
5.3.3 Integrasjon og Datahåndtering

Applikasjonen benytter Azure DevOps for versjonskontroll og kontinuerlig integrasjon/deploy (CI/CD), sikrer at kodeendringer blir håndtert på en effektiv måte og applikasjonen oppdateres automatisk med minimal menneskelig intervensjon. Azure DevOps fungerer også som en sentral hub for prosjektstyring, sporing av oppgaver og sprinter, som bidrar til en strømlinjeformet utviklingsprosess.

Dette systemet tillater ikke bare rask utvikling og deployering av nye funksjoner, men også en høy grad av kontroll og oversikt over applikasjonens levetid og ytelse. Ved å benytte moderne

skybaserte løsninger og automatiserte arbeidsflyter, sikrer arkitekturen at applikasjonen kan vokse og tilpasse seg etter behov uten å gå på kompromiss med kvaliteten på tjenestene som tilbys.

5.4 Filstruktur



Figur 9: Overordnet mappestruktur i prosjektet

Prosjektets arkitektur er organisert i tråd med moderne programvareutviklingsstandarder, og ansvarsområder og funksjoner for hver komponent er tydelig definert. Filstruktur er designet for å håndtere ulike aspekter av prosjektet på en isolert og modulær måte. Her er en oversikt over de viktigste komponentene og mappene i prosjektet:

ClientApp	Denne mappen er viet til brukergrensesnitt (UI)-utviklingen, inneholder klient-siden ressurser og adopterer en Single Page Application (SPA) arkitektur.
node_modules	Denne mappen inneholder alle tredjepartspakker som er installert via Node.js pakkeforvalter, npm.
public	Inneholder applikasjonens offentlige statiske filer, som favicon og index.html.
src	Dette er hovedmappen som inneholder kildekoden, og inkluderer følgende undermapper
components	Inneholder gjenbrukbare React-komponenter og øker effektiviteten i utviklingsprosessen ved å redusere kodegjentakelse.
pages	Inneholder websider som brukerne vil samhandle med direkte
images	Holder bilder som vil bli brukt i applikasjonen.
package.json	En konfigurasjonsfil som spesifiserer applikasjonens avhengigheter og script som kan kjøres med npm.
Controllers	En mappe som inneholder server-siden logikk for håndtering av forespørsler, i tråd med ASP.NET Core MVC eller Web API arkitektur.
Pages	En mappe som adopterer ASP.NET Core's Razor Pages arkitektur og inneholder server-rendered sider.
Properties	Inneholder filer som spesifiserer runtime settings og konfigurasjoner for ulike miljøer.
wwwroot	En mappe tilgjengelig direkte av nettlesere, inneholder statiske filer for applikasjonen (CSS, JavaScript, bilder, etc.).
***.csproj og *.sln**	.NET løsnings- og prosjektfiler, definerer prosjektets konfigurasjon og avhengigheter.
.gitignore	Definerer filer og mapper som versjonskontrollsystemet git skal ignorere.
appsettings.json	Inneholder konfigurasjonsinnstillinger for applikasjonen og tilpasser seg ulike utviklings- eller produksjonsmiljøer.
README.md	Et dokumentasjonsfil som presenterer prosjektet og inneholder informasjon om installasjonsprosessen, bruk osv.

Tabell 5: Viktigste komponentene og mappene i prosjektet

Denne strukturelle organisasjonen øker prosjektets bærekraft og kodens lesbarhet, samtidig som den muliggjør samarbeid mellom teammedlemmer under utviklingsprosessen. Adopsjonen av en modulær tilnærming bidrar til å minimere avhengigheter mellom komponenter og opprettholde en renere kodebase.

6. Testing

I dette kapittelet vil vi utforske de ulike testmetodene som er anvendt for å sikre funksjonalitet, pålitelighet og brukervennlighet av vår applikasjon. Fra kode testing til brukerinteraksjoner, vil vi dekke hvordan hver testingstilnærming bidrar til et robust og effektivt sluttprodukt.

6.1 Kode Testing

For å sikre høy kvalitet og pålitelighet i applikasjonen, har vi implementert en omfattende teststrategi ved hjelp av moderne testingrammeverk og verktøy. Backend-testene våre er utviklet med Xunit og Moq for å tilby grundige enhetstester og mocking-funksjonaliteter. Xunit ble valgt for sin fleksibilitet og støtte for parallellkjøring av tester, noe som akselererer testprosessen. Det er et populært rammeverk innen .NET-økosystemet for skriving av reproduserbare tester som kan kjøre uavhengig av hverandre, noe som bidrar til mer effektiv feilsøking og validering av kode. Moq brukes for å enkelt opprette mock-objekter for avhengigheter, som IHttpClientFactory, IConfiguration, og IAzureDevOpsService, noe som gjør det mulig å teste controllerens logikk isolert fra eksterne tjenester og konfigurasjoner. Dette frameworket er spesielt verdsatt for sin evne til å forenkle opprettelsen av disse mock-objektene, som er avgjørende for å isolere testene fra eksterne avhengigheter (Xunit, 2022).

I tillegg til backend-testene, har vi strukturert våre SpecFlow- og Playwright-tester for end-to-end testing av applikasjonen. SpecFlow benyttes for å brobygge mellom forretningskrav og teknisk implementasjon, ved å tilrettelegge for Behavior-Driven Development (BDD). Dette rammeverket hjelper teamet med å formulere testscenarioer på et språk som er lett forståelig, noe som sikrer at testene er i tråd med forretningsmålene og lett kommuniseres over faggrupper. Playwright, på sin side, brukes for automatisert nettlesertesting på tvers av alle moderne nettlesere. Playwright er kjent for sin høye ytelse og støtte for testing på tvers av nettleserplattformer, noe som gjør det til et ideelt verktøy for å sikre konsistent brukeropplevelse på tvers av forskjellige nettlesermiljøer. Dette sikrer at vår applikasjon fungerer som forventet for sluttbrukerne, uavhengig av deres valg av nettleser (SpecFlow, 2022; Playwright, 2022).

Valget av disse testrammeverkene og -verktøyene gjenspeiler vår forpliktelse til å opprettholde høyeste standarder for kodekvalitet og applikasjonssikkerhet. Ved å kombinere enhetstesting med end-to-end testing, sikrer vi at alle aspekter av applikasjonen er grundig validert før utrulling, noe som bidrar til en feilfri brukeropplevelse.

6.2 Ekspert Testing

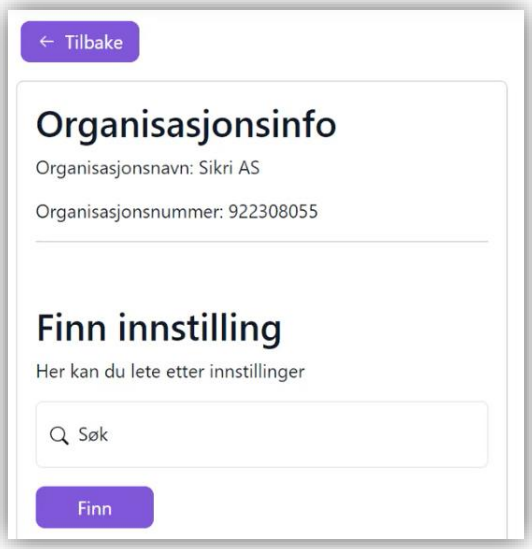
Eksperttesting innebærer å evaluere et design mot retningslinjer for brukervennlighet, prinsipper fra felt som kognitiv psykologi og menneske-maskin-interaksjon, og vurderinger basert på ekspertisen og tidligere erfaringer til gjennomgangspersonen i feltet. Vanligvis resulterer eksperttesting i en detaljert skriftlig dokumentasjon med anbefalinger; ekspertise innen UX utvikles gjennom betydelig eksponering for reell brukeratferd og UX-forskning, heller enn et bestemt antall års erfaring (Nielsen Norman Group, 2018).

For å sikre kvaliteten på prosjektet gjennomførte gruppen en serie med eksperttester med to personer, hvor den ene hadde betydelig erfaring med UX/UI-design og den andre hadde omfattende konsulentkompetanse. Begge ekspertene ga verdifulle tilbakemeldinger på systemet og veiledet gruppen mot et "Pixel-Perfect" design. Pixel-perfect innebærer å designe med piksler i tankene og implementere nøyaktig samme design på skjermen, ned til hver eneste piksel, samtidig som man sørger for responsiv tilpasning til andre enheter (Stetsenko, u.å.). Imidlertid var produkteierens forventning hovedsakelig knyttet til funksjonalitet, ikke nødvendigvis et perfekt design. Derfor måtte gruppen finne en balanse mellom "pixel-perfect" design og produkteierens ønsker. Ved å ha intervjuer med ekspertene fikk vi veiledning om systemflyt, identifiserte problemområder, mottok anbefalinger og beste praksis for løsninger, samt fikk en liste over mulige brukervennlighetsproblemer mens de gjennomgikk systemflyten. I *appendiks 10* presenteres intervjuguide for eksperttesting.

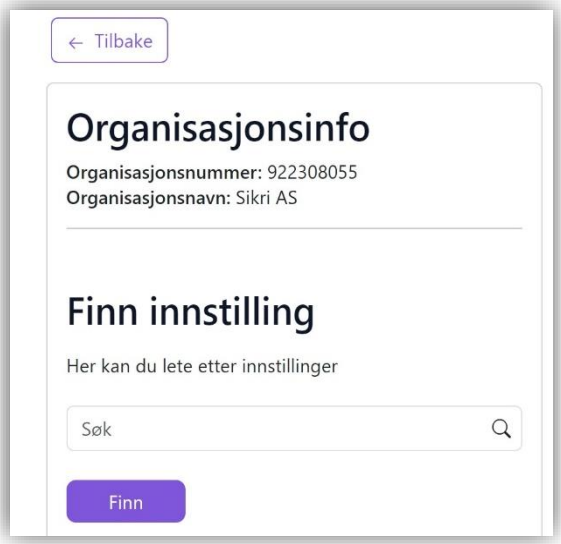
Figur 11: Forrige ikon

Figur 10: Nytt ikon

For eksempel, som vist i figur 10, justerte gruppen ikonene i henhold til tilbakemeldingene fra ekspertene om formularkortene, da det opprinnelige ikonet ikke passet til prinsippet om "Familiarity" innen UX. Det opprinnelige ikonet ble oppfattet som et sikkerhetsikon, men ble endret til et innstillingsikon.



Figur 12: Forrige tilbakeknapp



Figur 13: Ny tilbakeknapp

Gruppen gjorde også en annen endring. Designet som vist i figur 12 ble endret til figur 13 Tilbakeknappen var forvirrende for brukeropplevelsen. Dette var problematisk med prinsippet om "Affordance", siden knappen hadde samme farge som primærfargen, noe som førte til at brukerne trykket på den først. En knapp burde ha større betydning, som for eksempel "Finn" knappen.

6.3 Bruker Testing

Bruker testing er en kritisk evalueringsmetode der faktiske brukere tester et produkt eller system for å identifisere problemer og foreslå forbedringer innen funksjonalitet, brukervennlighet og tilfredshet. Denne metoden er fundamentalt viktig da den tilbyr direkte innsikt fra målgruppen, og kan avdekke problemer som kanskje ikke blir oppdaget gjennom interne eller automatiserte tester (Benyon, 2019).

I QuickFix-applikasjonens prosjekt ble bruker testing benyttet for å forstå hvordan ekte brukere interagerer med applikasjonen, noe som er essensielt for å identifisere og korrigere brukergrensesnittproblemer, forbedre navigasjon, og sikre at applikasjonen møter brukernes reelle behov. Den første versjonen av brukergrensesnittet, som vist i Figur 13, hadde bare en enkel "Logg inn" knapp. Tilbakemeldinger fra testing med to brukere viste behovet for mer informasjon og en mer intuitiv tilnærming, noe som førte til en endring i grensesnittet. Spørsmålene som ble benyttet under brukertestingene er dokumentert i *Appendiks 11 – Brukertest*

intervjuguide, hvor detaljene rundt interaksjonene og spørsmålene som ble stilt til brukerne er tilgjengelige.

Figur 14: Forrige innloggingskort

Figur 15: Nytt innloggingskort

Basert på brukernes tilbakemeldinger ble grensesnittet revidert for å inkludere ytterligere informasjon om at QuickFix raskt løser kundeproblemer, sammen med en modifisert innloggingsknapp som inviterer brukerne til å "Logg inn med din konto". Dette er illustrert i Figur 15. Disse endringene var ment for å gjøre applikasjonens formål klarere og innloggingsprosessen mer intuitiv, noe som forbedrer den generelle brukeropplevelsen ved å gjøre funksjonaliteten mer tilgjengelig og forståelig.

Endringene i brukergrensesnittet reflekterer prosjektteamets respons på direkte brukerinput og deres forpliktelse til å skape en brukervennlig og effektiv digital plattform. Det reviderte designet tar sikte på å forbedre brukernes førsteinntrykk og støtte en sømløs brukeropplevelse, noe som viser hvordan bruker testing direkte påvirker utviklingen av applikasjonen for bedre å møte brukernes behov og forventninger.

7. Prosjektgjennomføring

Dette avsnittet tar for seg prosjektgruppens metodiske gjennomføring og administrasjon av prosjektet. Det fokuseres spesielt på hvordan scrum-rammeverket er benyttet for å fasilitere arbeidsflyten, med en spesifikk vektlegging på produkt- og sprint-backlogger, samt organiseringen og gjennomføringen av arbeidet i henhold til disse. Videre vil avsnittet dekke tilnærmingen til systematisk analyse, applikasjonsdesign, sprintinndeling, testing og integrering av resultater.

7.1 Scrum-implementering

Gruppen har benyttet Scrum-metodikken primært fordi den støtter rask tilpasning til endringer og fremmer en iterativ utviklingsprosess, noe som er avgjørende i komplekse prosjektomgivelser. Gjennom implementeringen av Scrum har vi kunnet forbedre arbeidsflyten betydelig og sikre jevn fremgang i prosjektet. Metodikken har vært spesielt nyttig i håndtering av produkt- og sprint-backlogger, som har hjulpet oss med å prioritere oppgaver og tilpasse oss raskt til prosjektets skiftende behov. Videre har daglige Scrum-møter, sprintevalueringer og retrospektiver spilt en sentral rolle i å opprettholde høyt engasjement og transparens i teamet, samt å identifisere og adressere utfordringer effektivt. Valget av Scrum ble drevet av behovet for en fleksibel, men strukturert tilnærming til prosjektstyring som kunne tilrettelegge for kontinuerlig forbedring og effektiv kommunikasjon innad i teamet.

7.1.1 Sprint Planning

Sprintplanleggingen var avgjørende i vårt prosjekt for softwareutvikling, gjennomført i nært samarbeid med veilederen fra bedriften. Denne planleggingsfasen omfattet utforming av en omfattende mal som tydelig definerte prosjektets fremdrift og de tilhørende oppgavene. Ved starten av hver ny sprint, og ved avslutningen av den forrige, ble det lagt stor vekt på å detaljere spesifikke brukerhistorier og de relaterte oppgavene. Dette sikret en velstrukturert tilnærming til prosjektets utførelse.

Videre benyttet vi oss av MoSCoW-metoden for prioritering for å oppnå bedre oversikt og effektivitet i håndteringen av prosjektets oppgaver (Schwaber & Sutherland, 2020). Denne metodikken var instrumental for å identifisere kritiske oppgaver og deres utførelsesrekkefølge, noe som fremhevet betydningen av nøye sprintplanlegging for å oppnå prosjektets mål (Rubin, 2012). En slik strategisk og veloverveid planleggingsprosess er avgjørende for suksessen i alle softwareutviklingsprosjekter, hvor målet er å maksimere produktiviteten og sikre prosjektets fremgang på en målrettet og effektiv måte (Cohn, 2006).

7.1.2 Daily Scrum

Daglige Scrum-møter spilte en sentral rolle i å holde prosjektet vårt på sporet, spesielt gitt vårt hybride arbeidsmiljø. To dager i uken var teamet samlet på kontoret, hvor vi gjennomførte disse møtene ansikt til ansikt, noe som la grunnlaget for en sterk teamfølelse og direkte kommunikasjon (Sutherland & Schwaber, 2020). De resterende tre dagene, når vi arbeidet hjemmefra, ble daglige Scrum-møter utført via Zoom. Denne tilnærmingen sikret at alle teammedlemmer forble engasjerte og motiverte, uavhengig av deres fysiske lokasjon (Moe, Dingsøyr, & Dybå, 2008).

Videre bidro disse møtene til å holde oppgavene organiserte og lette å følge. Ved å dele daglige oppdateringer, kunne vi raskt identifisere eventuelle hindringer og tilrettelegge for umiddelbare løsninger, noe som var avgjørende for å opprettholde prosjektets fremdrift (Kniberg & Skarin,

2010). Daglige Scrum-møter viste seg derfor å være uvurderlige for både teamets moral og den generelle prosjektforvaltningen, ved å skape et miljø der kontinuerlig forbedring var mulig.

7.1.3 Sprint Review og Retrospekt

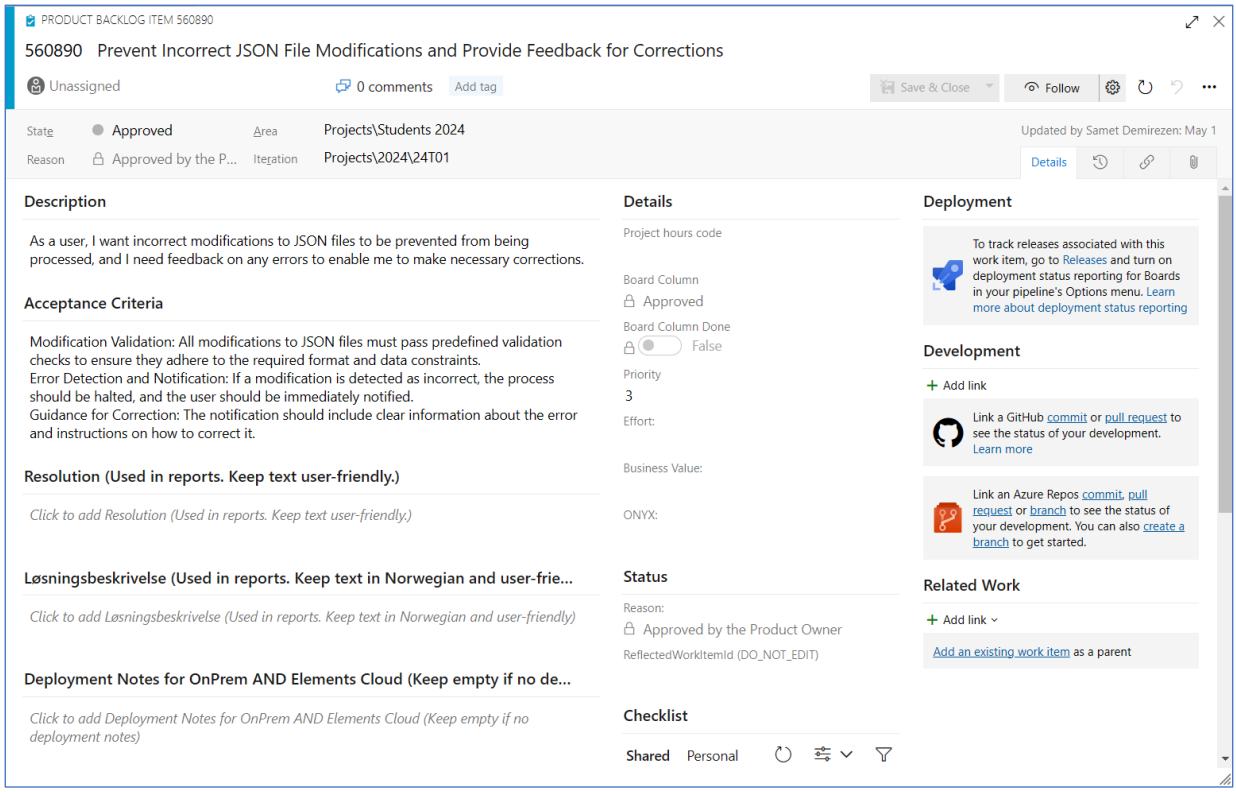
Sprint Review og Retrospekt-møter ble nøye gjennomført på den siste dagen av hver sprint, og fungerte som en integrert del av vår agile metodikk (Schwaber & Sutherland, 2020). I disse møtene fokuserte vi på å svare på vanlige spørsmål om sprintens resultat, hva fungerte, hva fungerte ikke, og hva kan gjøres bedre. Hver deltager hadde muligheten til å dele sine meninger og refleksjoner, noe som fremmet et miljø av åpenhet og kontinuerlig forbedring (Derby, Larsen, & Schwaber, 2006).

Denne prosessen førte til en dyptgående innsikt i våre feilgrep og utfordringer, og gjorde det mulig for oss å identifisere og adressere problemer på en strukturert måte. For å sikre at læringspunktene og beslutningene fra disse møtene ble bevart, førte vi nøye logg over dem i et "møtereferat"-dokument. Dette bidro til å holde alle teammedlemmer informert og sikret at tiltakene ble gjennomført i påfølgende sprinter (Kniberg & Skarin, 2010). Å opprettholde en detaljert dokumentasjon av sprint review og retrospekt-møter er essensielt for å sikre kontinuerlig forbedring og suksess i agile prosjekter, ved å skape et grunnlag for læring og tilpasning gjennom hele prosjektets levetid.

7.2 Produkt Backlogg

Et produktbacklog er en prioritert liste avledet fra produktets veikart og dets krav, som beskriver oppgavene for utviklingsteamet. Denne listen oppdateres kontinuerlig etter hvert som prosjektet skrider frem, og opprettholdes i en fleksibel struktur for å sikre at teamet oppnår sine mål. Produktlederen gjennomgår og justerer jevnlig prioriteringene innenfor denne listen for å sikre effektiv arbeidsflyt og samsvar med prosjektmålene (Radigan, u.å.).

Backloggen ble nøye forberedt basert på brukerhistoriene som hadde blitt utviklet, med hver enkelt prioritert etter sin betydning for prosjektet. Teamet tok seg også tid til å definere detaljerte akseptansekriterier for hver oppgave, noe som var avgjørende for å sikre at alle nødvendige forhold ble oppfylt før en brukerhistorie ble ansett som fullført. Disse akseptansekriteriene spilte en nøkkelrolle i prosjektledelsen og bidro til å klargjøre målene for hver oppgave. Et eksempel på hvordan disse akseptansekriteriene ble visualisert og anvendt i praksis kan sees i Figur 16.

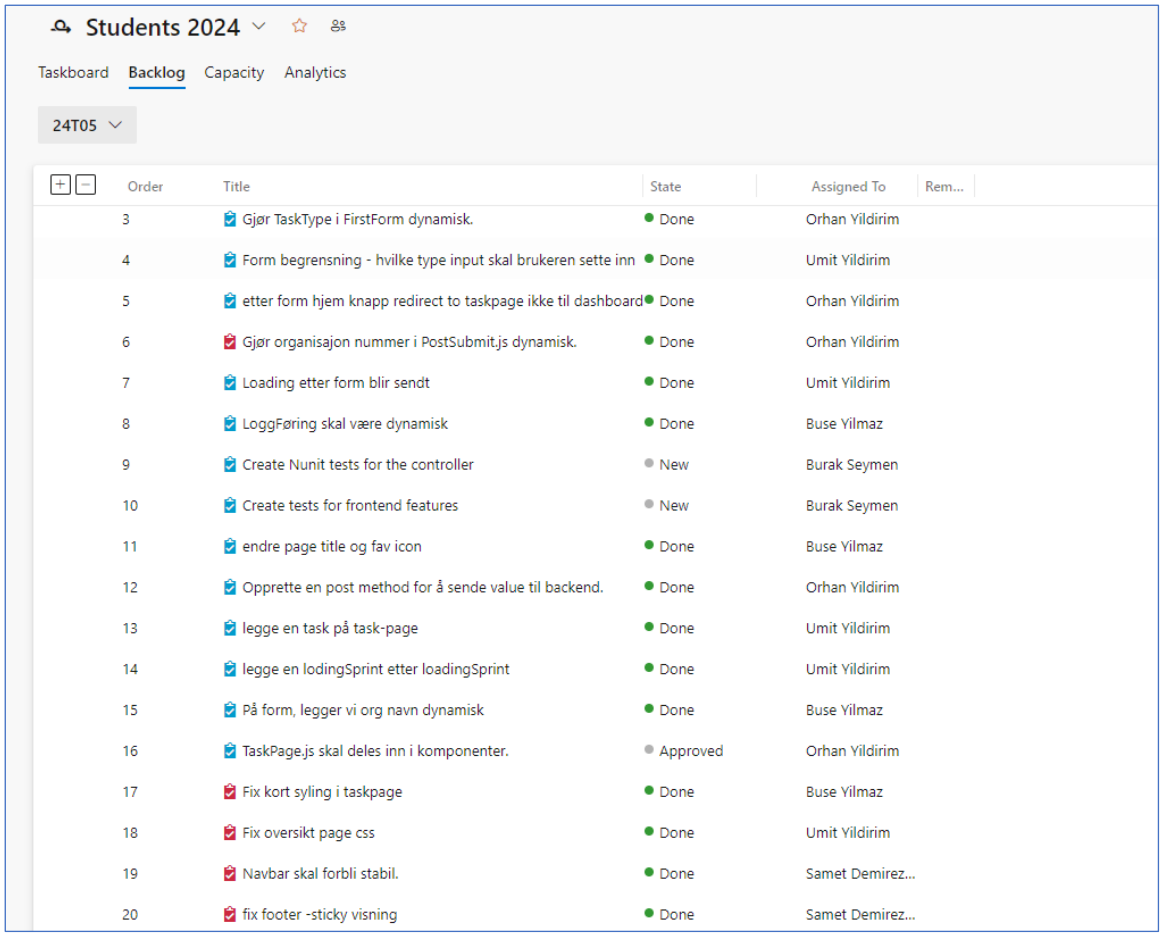


Figur 16: Produkt backlogg

7.3 Sprint Backlogg

I et Agile prosjektstyringsmiljø er Sprint Backlog en kritisk komponent som inneholder elementer Scrum-teamet har som mål å fullføre i løpet av en sprint. Denne listen vanligvis omfatter nye funksjoner, feilrettinger, teknisk gjeld, og kunnskapsinnhenting, som er en del av et større produktveikart (Ramsøy, 2022). Sprint Backlog er et levende dokument som kontinuerlig oppdateres, noe som gir teamet mulighet til å administrere arbeidsmengden og måle fremgang mot sprintens mål, og slik støtter den fleksibilitet og kontinuerlig forbedring gjennom utviklingsprosessen (Raeburn, 2024).

I vårt prosjekt, for eksempel, før hver sprint, utførte teamet vårt planleggingen som vist i Figur 17: Sprint 5 backlog. Oppgavestatus ble dynamisk håndtert av teammedlemmene og kategorisert som 'To Do', 'In Progress', og 'Done'. Dette sikret løpende oppdatering av fremgang og transparent tilbakemelding til interessenter, og teamet lyktes i å produsere verdifulle iterasjoner av produktet ved hver sprintslutt.



Order	Title	State	Assigned To	Rem...
3	Gjør TaskType i FirstForm dynamisk.	Done	Orhan Yildirim	
4	Form begrensning - hvilke type input skal brukeren sette inn	Done	Umit Yildirim	
5	etter form hjem knapp redirect to taskpage ikke til dashboard	Done	Orhan Yildirim	
6	Gjør organisajon nummer i PostSubmit.js dynamisk.	Done	Orhan Yildirim	
7	Loading etter form blir sendt	Done	Umit Yildirim	
8	LoggFøring skal være dynamisk	Done	Buse Yilmaz	
9	Create Nunit tests for the controller	New	Burak Seymen	
10	Create tests for frontend features	New	Burak Seymen	
11	endre page title og fav icon	Done	Buse Yilmaz	
12	Opprette en post method for å sende value til backend.	Done	Orhan Yildirim	
13	legge en task på task-page	Done	Umit Yildirim	
14	legge en lodingSprint etter loadingSprint	Done	Umit Yildirim	
15	På form, legger vi org navn dynamisk	Done	Buse Yilmaz	
16	TaskPage.js skal deles inn i komponenter.	Approved	Orhan Yildirim	
17	Fix kort syling i taskpage	Done	Buse Yilmaz	
18	Fix oversikt page css	Done	Umit Yildirim	
19	Navbar skal forbli stabil.	Done	Samet Demirez...	
20	fix footer -sticky visning	Done	Samet Demirez...	

Figur 17: Sprint 5 backlogg

7.4 Sprinter

I denne delen av prosjektet gjennomgår vi sprintprosessen, som er en kjernekomponent i vår agile utviklingsmetodikk. Hver sprint representerer en dedikert tidsramme hvor spesifikke oppgaver og mål prioriteres og gjennomføres. Dette kapittelet vil detaljert beskrive hver sprint, inkludert forberedelsene, de primære oppgavene som ble håndtert, og de spesifikke resultatene som ble oppnådd. Vi vil også reflektere over læringspunktene og justeringene gjort underveis, som har vært avgjørende for den kontinuerlige forbedringen av prosjektet og teamets effektivitet.

7.4.1 Pre-Sprint (5. januar – 18. januar)

I begynnelsen av pre-sprint-perioden ble tiden brukt til å lære og forsøke å forstå prosjektet, bli kjent med kollegaer på kontoret og organisere ankomstdager og -tider. I løpet av denne perioden ble det fokusert på en rekke nøkkeloppgaver for å sikre en solid start på prosjektet. For det første ble det bestemt hvilke teknologier og verktøy som skulle brukes i prosjektet, og utviklingsmiljøene ble installert på datamaskinene. Videre ble det utformet en detaljert

prosjektplan som omfattet alle faser og leveranser. Denne planen inkluderte en analyse av eksisterende løsninger som adresserte lignende utfordringer, og tilbød verdifull innsikt i beste praksis og innovasjoner i feltet. Det ble også gjennomført en omfattende litteraturgjennomgang for å sikre at teamet var fullstendig oppdatert på relevante teorier og forskning.

I løpet av pre-sprint-fasen spilte møtene en avgjørende rolle i diskusjonene om prosjektets potensielle mål og omfang. Disse samlingene var sentrale for å klargjøre prosjektets visjon og sikre enighet blant alle teammedlemmer. Gjennom denne forberedelsesfasen var teamet engasjert i å strukturere en velorganisert fordeling av ansvar og nøye planlegging av sprintene, noe som bidro til en effektiv arbeidsflyt og samarbeid. Identifisering og planlegging av potensielle risikoer var også en kritisk komponent, og teamet utarbeidet en risikoanalyse for å forutse og effektivt håndtere potensielle utfordringer.

7.4.2 Sprint 1 (19. januar – 1. februar)

I løpet av sprint 1 oppsto det kommunikasjonssvikt med Kristoffer, noe som førte til problemer i samarbeidet og påvirket effektiviteten til teamet. For å løse dette, bestemte vi oss for å etablere en fast møtedag, noe som bidro til å forbedre planleggingen og kommunikasjonen. Denne strukturerte tilnærmingen hjalp til med å klargjøre forventningene, sikre at alle teammedlemmer var informert om fremdriften, og opprettholde en konsistent arbeidsflyt. Teamet fokuserte intensivt på å forstå problemområdet. Vi gjennomførte flere dybdeintervjuer med kundesenteret for å samle relevant og verdifull informasjon som kunne veilede utviklingsprosessen.

For å generere ideer og finne optimale løsninger, arrangerte vi en omfattende brainstorming-sesjon. Alle teammedlemmene bidro aktivt med skisser og forslag, og gjennom samarbeid og diskusjon, klarte vi å identifisere de mest hensiktsmessige tilnærmingene. Dette la et solid fundament for prosjektets videre fremdrift. Videre brukte vi betydelig tid på å planlegge designfasen for å være godt forberedt til sprint 2. Denne planleggingen inkluderte opprettelsen av detaljerte oppgavelister, milepæler og tidslinjer, noe som bidro til å sikre en strukturert og målrettet utviklingsprosess fremover. Vi diskuterte også og bestemte hvilke verktøy og teknologier som ville støtte prosjektets behov best. Teamet startet med å sette opp de nødvendige utviklingsmiljøene og sørget for at alle hadde tilgang til de nødvendige ressursene. I tillegg utarbeidet vi en risikoplan for å identifisere potensielle utfordringer tidlig og utvikle strategier for å håndtere dem effektivt. Dette omfattende forberedelsesarbeidet var avgjørende for å sikre en effektiv arbeidsflyt og et godt samarbeid i de kommende sprintene, og etablerte en klar visjon og retning for prosjektet.

7.4.3 Sprint 2 (2. februar – 15. februar)

I sprint 2 har gruppen startet arbeidet med å skrive de innledende delene av rapporten, for eksempel sentrale valg og beslutninger som er tatt for prosjektet. For å sikre kvalitet i prosjektet, ble det enighet om å implementere en kodestandard i samarbeid med Sikri. Dette tiltaket er ment

å forbedre forståelsen av koden under videreutviklingsfasen og øke kvaliteten på det endelige produktet.

Basert på funnene fra intervjuene med brukerne og produkteieren, utarbeidet vi en MoSCoW-tabell for å tydeliggjøre prioriteringene i prosjektet. Hensikten med dette var å bedre estimere tidsbruken og unngå unødvendig kompleksitet. Etter prioriteringen av funksjoner i applikasjonen, samarbeidet gruppen om å lage en wireframe-prototype på Figma. Dette ble gjort for å legge til rette for en smidig utviklingsprosess. Alle forberedelser ble fullført for å kunne starte utviklingen.

Gruppen utfordret seg selv til å begynne med frontend-utviklingen så tidlig som mulig i sprint 2, men dette viste seg å være utfordrende. Målet for sprint 3 er å gradvis utvikle applikasjonen steg for steg i stedet for å prøve å gjennomføre alt på én gang. Dette vil innebære å dele oppgavene i mindre deler for å sikre en mer effektiv utviklingsprosess.

7.4.4 Sprint 3 (16. februar – 29. februar)

I Sprint 3 siktet vi på å utvikle applikasjonen gradvis, trinn for trinn, og gruppen viste god samarbeidsevne i opprettingen av oppgaver. Dette muliggjorde en mer effektiv utviklingsprosess ved å dele oppgavene inn i mindre, håndterbare deler.

Gjennom sprinten begynte teamet å designe sider for applikasjonen, grunnleggende funksjoner som gettere og settere ble opprettet i backend, og viktige komponenter som "dashboard" og "taskpage" begynte å bli utviklet i frontend. CSS-konflikter mellom sidene nødvendiggjorde en reorganisering av CSS-filene. Problemer ble løst ved å opprette individuelle komponenter for hver side, og denne tilnærmingen understreket viktigheten av modulær utvikling i håndteringen av både funksjonalitet og brukergrensesnitt, noe som bidro til å opprettholde en konsistent brukeropplevelse gjennom hele applikasjonen.

7.4.5 Sprint 4 (1. mars – 14. mars)

I løpet av den fjerde sprinten i vårt prosjekt rettet vi fokus mot å utvide og forbedre funksjonaliteten i både backend og frontend. På backend implementerte vi grunnleggende GET- og POST-metoder, som betraktelig forbedret systemets kapasitet til å behandle data både effektivt og responsivt.

I grensesnittet introduserte vi et nytt skjema som aktiveres når brukere engasjerer seg med oppgaver på oppgavesiden. Vi forbedret kontinuerlig strukturen til dette skjemaet og prioriterte å sikre en sømløs integrasjon mellom frontend og backend. Dette innebar en feilfri overføring av data fra backend til frontend og en uforstyrret overførsel av endringer fra frontend-filer til backend.

Videre arbeid med brukergrensesnittet innebar forbedringer av CSS for å heve både estetikk og brukervennlighet. Vi fortsatte også arbeidet med å utarbeide rapportdelen av vår

bacheloroppgave, responderte på tilbakemeldinger fra vår veileder, og distribuerte oppgaver for å fasilitere skriving av nye seksjoner.

Denne sprinten ble dedikert til å styrke og konsolidere applikasjonens funksjonalitet og brukerinteraksjoner, med et vedvarende fokus på tekniske forbedringer og utviklingen av en robust applikasjon.

Videre diskusjoner i sprinten omhandlet planleggingen for den kommende Sprint 5, hvor vi vil arbeide for å styrke integrasjonen mellom frontend og backend samt forbedre sikkerhetsaspektene ved koden.

7.4.6 sprint 5 (15. mars – 28. mars)

I Sprint 5 gikk arbeidet hovedsakelig ut på applikasjonsutvikling. I denne sprinten arbeidet vi med å få frontend og backend til å fungere sammen og være kompatible. Selv om integreringsarbeidet tok litt lengre tid enn forventet, førte det ikke til noen større problemer. Samtidig ble det i denne sprinten brukt mye tid på å begrense brukerinput i applikasjonens frontend, noe som er en av de viktigste funksjonene. Dette var en viktig funksjon fordi en feil verdi inntastet av brukeren kan føre til at tjenesten ikke fungerer eller fungerer feil, noe som kan føre til store problemer med hensyn til kundetilfredshet. Samtidig ble noen små feil i applikasjonen også rettet opp i denne perioden.

I tillegg ble det i denne sprinten gjennomført bruker- og eksperttesting for å sikre og forbedre produktkvaliteten. Under brukertesting fikk vi viktige tilbakemeldinger fra faktiske brukere, og vi mottok ideer om hvordan en bedre brukeropplevelse kunne oppnås. Eksperttesting var også nyttig for å forstå hvordan vi kunne tilby en bedre brukeropplevelse, og vi implementerte resultatene av disse to testene i den neste sprinten. For mer detaljert informasjon kan det ses *6.2 Ekspert testing* og *6.3 Bruker testing* seksjoner.

Til slutt forberedte vi også dokumentasjon i denne sprinten for å øke anvendeligheten og utviklingsmulighetene til applikasjonen *(Appendiks 13*). Denne dokumentasjonen diskuterte hvordan applikasjonen kunne installeres, videreutvikles og noen viktige systeminnstillinger. Gjennom hele sprinten fortsatte vi også med å jobbe med rapportskriving. I denne sprinten sikter vi mot å eliminere tapet av motivasjon forårsaket av store oppgaver ved å definere dem i mindre skalaoppgaver.

7.4.7 Sprint 6 (29. mars – 11. april)

I sprint 6 var fokuset rettet mot kvalitetssikring, hvor formålet var å sikre oppfyllelse av alle nødvendige krav for å oppnå en høy kvalitet på prosjektet. En sentral del av denne innsatsen involverte utarbeidelse av dokumentasjon for å sikre at koden ville være tilstrekkelig forståelig og vedlikeholdbar i videreutviklingsfasen. Denne tiltaksplanen ble etablert som et tidligere definert krav. Videre ble våre evalueringer basert på både brukertesting og ekspertvurdering etter

fullførelsen av applikasjonen. Resultatene av disse evalueringene indikerte at vår applikasjon ble utviklet i samsvar med de opprinnelige forventningene.

Kodetestingen, som var et sentralt aspekt for å sikre kvaliteten, ble gjennomført, og samtlige tester bestod, i samsvar med våre definerte kvalitetsstandarder. Kontinuerlige ukentlige møter med produkteieren ble avholdt, hvor vi diskuterte deres forventninger og vårt produkt. Vårt primære mål var å sikre leveranse av alle ønskede funksjonaliteter.

I sprint 7 møtte vi en utfordring knyttet til tidsstyring og fordeling av ressurser mellom arbeidet med rapporten og videre utvikling av applikasjonen. Planen var opprinnelig å jobbe parallelt med begge disse oppgavene, men dette viste seg å være vanskelig å gjennomføre. Som en respons på dette, vil vi i sprint 7 fokusere på å effektivisere vårt samarbeid og arbeid med rapporten. Videre vil sprint 7 være dedikert til deployment av applikasjonen, etter at vi har bekreftet at den er ferdigutviklet.

7.4.8 Sprint 7 (12. april – 25. april)

Hovedfokuset var på deployment av applikasjonen for Sprint 7. Vi opprettet både en Docker-fil og en YML-fil for å lette implementeringen, men vi møtte flere tekniske problemer som førte til betydelig tidstap. Spesielt var det autorisasjonsproblemer som hindret Docker-bildene fra å fungere korrekt. Disse utfordringene krevde omfattende feilsøking og justeringer. For å løse problemene, mottok vi teknisk støtte fra Sikri og samarbeidet tett med dem. Dette samarbeidet var essensielt for å overvinne hindringene og sikre at implementeringen kunne fortsette som planlagt.

I tillegg til arbeidet med deployment, fokuserte vi også på å forbedre og justere rapporten basert på tilbakemeldinger fra veilederen. Vi gjennomførte flere revisjoner og satte som mål å fullføre rapporten i sprint 8. For å effektivisere dette arbeidet, hadde vi allerede ved slutten av sprint 6 forbedret vår arbeidsmetodikk ved å dele opp oppgavene i mindre, mer håndterbare deler og øke antallet fysiske møter. Dette bidro til å opprettholde motivasjonen og forbedre samarbeidet innen teamet. Noen oppgaver ble utført som teamarbeid, noe som bidro til en bedre dynamikk og økt produktivitet. Vi gjennomførte omfattende bruker- og eksperttesting for å sikre høy kvalitet på produktet. Tilbakemeldingene fra disse testene var uvurderlige og ble implementert umiddelbart, noe som resulterte i forbedret funksjonalitet og brukeropplevelse. Vi dokumenterte nøye alle endringer og forbedringer for å sikre at applikasjonen var robust og enkel å vedlikeholde. Samtidig fortsatte vi arbeidet med å forbedre frontend- og backend-integrasjonen. Dette inkluderte justeringer for å sikre at alle komponenter fungerte sømløst sammen. Teamet arbeidet også med å implementere tiltak for å begrense brukerinput i applikasjonens frontend, noe som var kritisk for å forhindre feil og sikre en stabil drift. Denne sprinten var avgjørende for å konsolidere tidligere arbeid og sikre at applikasjonen var klar for endelig deployment og videre testing i sprint 8.

7.4.9 Sprint 8 (26. april – 25. april)

I denne sprinten jobbet teamet intensivt med den rapporten. Basert på tilbakemeldingene vi mottok fra veilederen gjorde vi flere revisjoner, noe som hjalp oss med å korrigere dokumentene og sikre at alle seksjonene var konsistente og informative. Gjennom denne prosessen hadde vi som mål å fullføre rapporten, men vi måtte fortsette med å korrigere feil og gjøre tillegg og trekk helt til den siste dagen. Vi jobbet kontinuerlig med å gjøre rettelser, legge til nye avsnitt og eliminere feil for å sikre at dokumentet var høy kvalitet.

I tillegg til arbeidet med rapporten, forberedte vi oss til den prosjektpresentasjonen. Teamet samarbeidet om å utarbeide en overbevisende presentasjon som skulle oppsummere vårt arbeid og høydepunktene fra prosjektet. Vi øvde flere ganger for å sikre at alle teammedlemmer følte 2,

Dette arbeidet var svært viktig for å sikre at alle aspekter av prosjektet ble grundig dokumentert og reflekterte teamets innsats og resultater gjennom prosjektet. Vår dedikasjon til prosjektet, grundigheten i våre forberedelser og den effektive kommunikasjonen mellom gruppe medlemmene, veilederen og produkt eieren har blitt identifisert som nøkkelkomponentene som sikret en vellykket avslutning på vårt prosjekt.

7.5 Ressursallokering

Dette delkapittelet forklarer hvordan vi har styrt ressurser i prosjektet. Ressursallokering dreier seg om å tildele ressurser, som tid, budsjett og mannskap, til spesifikke prosjekter og sprinter for å maksimere produksjonen (Kaur et al., 2023). Siden i oppgaven brukte vi bare tid og mannskap de skal nevnes hvordan gruppen maksimert effektiviteten av de ressursene.

7.5.1 Tilgjengelige ressurser

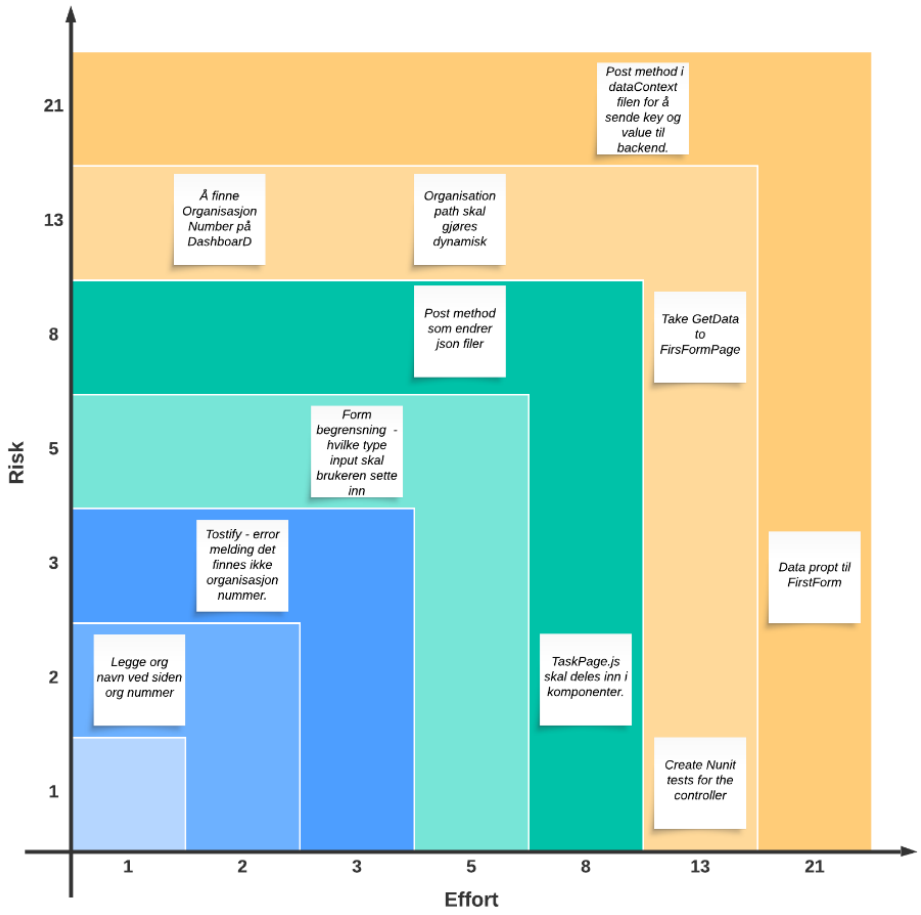
For å beregne tilgjengelige ressurser fra begynnelsen til slutten av prosjektet, bestemte gruppen hvor mange timer som kreves hver dag per person. Vi fikk informasjon fra fagansvarlig om at 5 timer per person per dag var påkrevde. Gruppen startet i pre-sprinten den 8. januar 2024, og etter å ha trukket fra helligdager satt gruppen igjen med 88 arbeidsdager. Med fem medlemmer i gruppen, beregnet vi totalt 2200 timer som tilgjengelige ressurser basert på timeforbruk. Disse tilgjengelige ressursene representerer den samlede tiden teamet har til rådighet for å utføre oppgavene i prosjektet. Det er den totale arbeidskapasiteten gruppen kan bruke for å oppnå prosjektmålene innenfor den angitte tidsrammen. Disse ressursene er avgjørende for planleggingen og gjennomføringen av prosjektet, og danner grunnlaget for tids- og ressursstyring underveis.

antall timer hver dag * antall dager i prosjektet * antall gruppemedlemmer = tilgjengelige ressurser 5 * 88 * 5 = 2200

Tabell 6: Tilgjengelige ressurser (Fredriksen et al.,2021)

7.5.2 Fibonacci agile estimering

Som tidligere nevnt under prosjektgjennomføringen i *kapittel 7*, gjennomførte gruppen sprint-planleggingsmøter. I disse møtene fastsatte gruppen utfordringsnivået og tidsestimatene for oppgavene ved hjelp av Fibonacci-sekvensen.



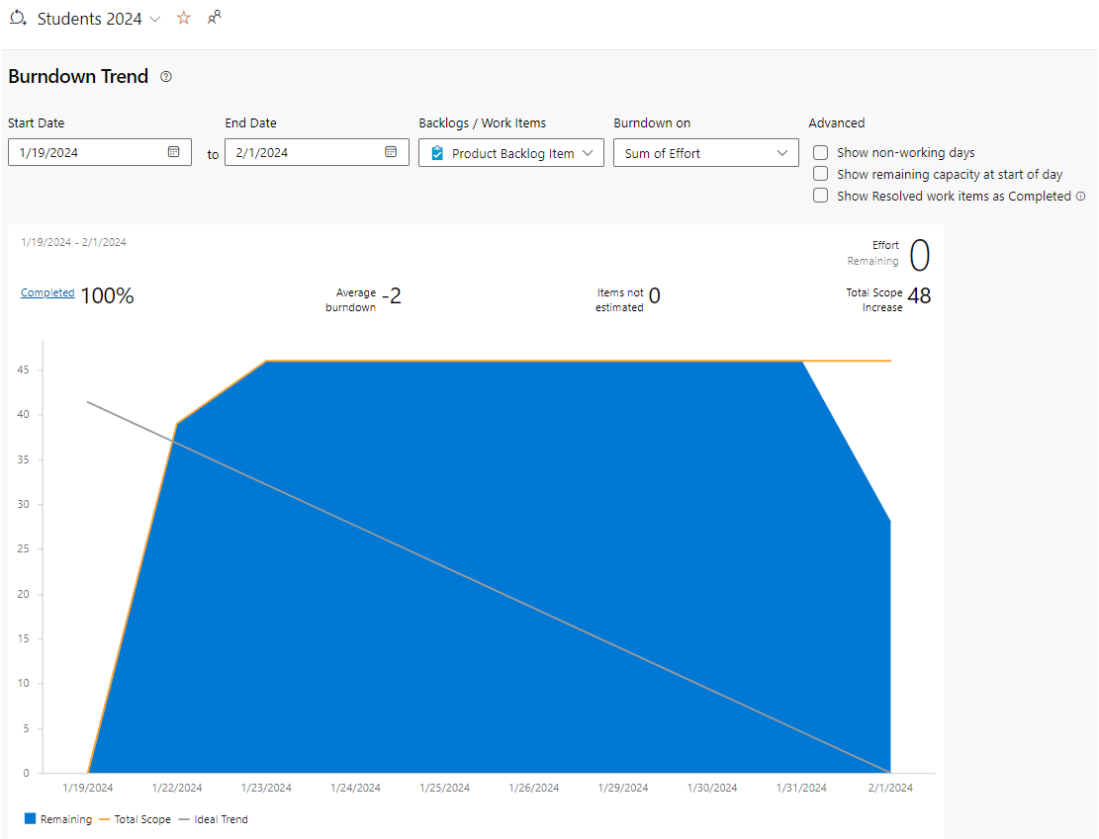
Figur 18: Fibonacci Agile estimeringstabell

I tillegg har gruppen utarbeidet en Fibonacci Agile Estimeringstabell, som vist i Figur 18 over. Denne tabellen er et eksempel for sprint 4. Som vanlig gjennomfører vi sprint-planleggingsmøter i begynnelsen av hver sprint. Gruppen samlet seg deretter for å definere oppgavene og plassere dem i tabellen ved å benytte seg av horisontale og vertikale verdier for innsats og risiko som referansepunkter. Det var av betydelig viktighet å involvere alle gruppemedlemmer i denne prosessen for å oppnå konsensus angående tidsestimeringer og risikonivåer. Hvert medlems

individuelle perspektiv bidro til å raffinere et mer presist og realistisk estimat av arbeidsomfanget nødvendig for å fullføre en brukerhistorie (Lucid Content, u.å.). Som et eksempel, samlet gruppen seg for å definere hva innsatsnivå 5 og risikonivå 8 innebar, og sammenlignet deres respektive verdier. Basert på denne vurderingen, ble oppgaven "Endre JSON-filer etter metode" plassert i krysningen mellom innsatsnivå 5 og risikonivå 8. Denne praksisen bidro til å øke oversikten over hver sprint, samt å fremme mer effektivt arbeid, samtidig som nøyaktigheten av estimatene ble forbedret.

7.5.3 Burn down chart

I denne delen presenteres et eksempel på en sprint ved bruk av Burn down chart, et av de vanligste verktøyene i Agile prosjektledelse, som vist i Figur 19. En burndown chart viser mengden arbeid som er fullført i en episk eller sprint, og det totale gjenværende arbeidet (Rehkopf, 2024). Diagrammet viser en sprintprosess som startet den 19. Januar 2024 og avsluttet den 2. Februar 2024. Bildet illustrerer konkret mengden arbeid som måtte fullføres gjennom sprinten, og fremgangen gjort av teammedlemmene. den 2. Februar 2024. Bildet illustrerer konkret mengden arbeid som måtte fullføres gjennom sprinten, og fremgangen gjort av teammedlemmene.



Figur 19: Burn down chart Sprint 2

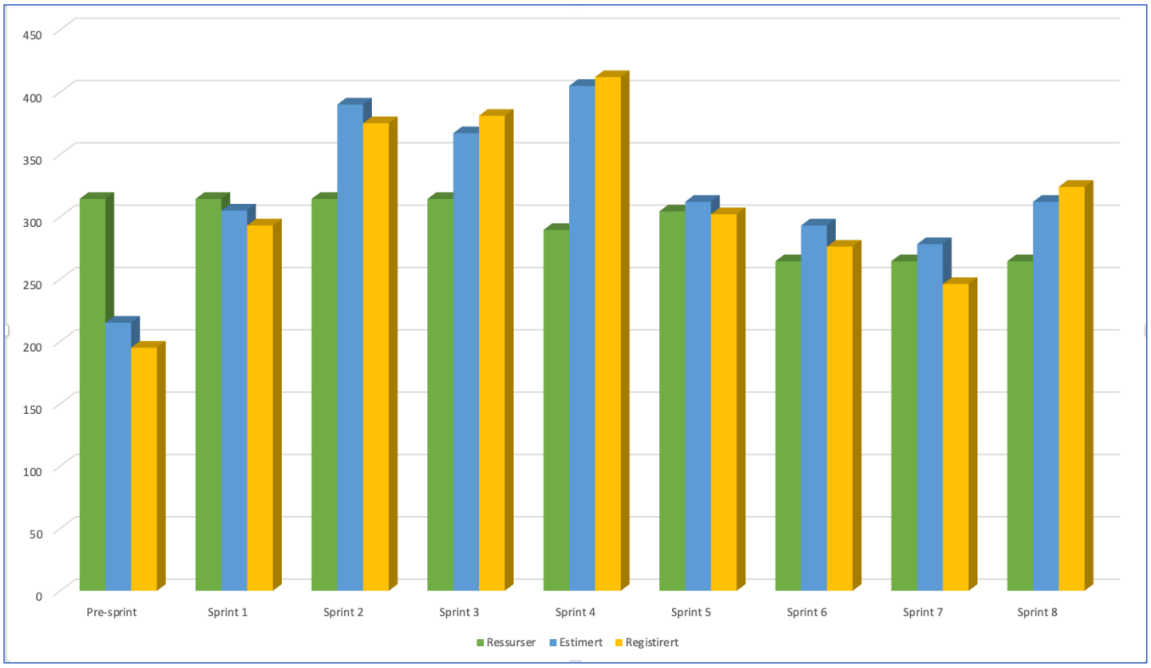
På figur 19 representerer det blå området gjenværende arbeidsmengde (Remaining), den oransje linjen det totale omfanget (Total Scope), og den grå linjen den ideelle fullføringstrenden (Ideal Trend). I begynnelsen av sprinten økte prosjektets omfang uventet; dette kan ha vært på grunn av tillegg av nye arbeidsoppgaver eller utvidelse av eksisterende oppgavers omfang. Imidlertid klarte teamet å håndtere denne ekstra belastningen gjennom sprinten og fullførte alle oppgavene før den forventede sluttdatoen.

I dette eksemplet indikerer betegnelsen "100% Completed" at teammedlemmene nådde sprintmålene innenfor den planlagte tiden, til tross for den økte arbeidsmengden. Uttrykket "Total Scope Increase 48" angir hvordan omfanget utvidet seg i løpet av sprinten. Burn down chart hjelper prosjektledere og teammedlemmer med å evaluere om sprinten ble fullført i tide, samt vurdere effekten av tillegg av arbeid og endringer i prioriteringer på prosjektet.

Figur 19 fungerer som et pedagogisk verktøy for å hjelpe studenter og nybegynnere innen prosjektledelse med å bedre forstå Agile metoder og sprintprosesser. Det viser også konkret de utfordringer som kan oppstå under prosjektsporing og evaluering, og metodene for å overkomme disse utfordringene.

7.5.4 Oversikt

Gruppen har benyttet et Excel-tabell for å håndtere ressursbruk. Tilgjengelige antall timer (grønn), estimater (blå) og faktiske registrerte (gull) timer er lagret i denne tabellen. Tilgjengelige ressurser, helligdager og dager medlemmene i gruppen har hatt fri, har blitt trukket fra for å bestemme dette. Selv om vi har forsøkt å regelmessig oppdatere og følge opp opptegnelsene, tok det lang tid å venne oss til systemet, og derfor er det noen avvik i registreringene. Gjennom alle sprintene er antall tilgjengelige, estimerte og registrerte timer vist i Figur 20.



Figur 20: Oversikt over ressursbruk

7.6 Risikohåndtering

I løpet av prosjektperioden har vi som team vedlikeholdt en detaljert risikologg som har fanget opp ulike hendelser som har påvirket fremgangen i prosjektet. Denne loggen, omtalt i *Appendiks 5*, har dokumentert hendelser, uten at noen av dem hadde en alvorlig risikokonsekvens. Totalt ti hendelser som har krevd vår oppmerksomhet og tiltak har blitt identifisert og loggført.

Blant disse hendelsene har vi registrert utfordringer som forsinkelser forårsaket av ekstreme værforhold, sykdom innad i teamet, og personlige forpliktelser som har krevd en omorganisering av møter og oppgaver. Selv om vi ikke har støtt på utfordringer med spesifikke tekniske kompatibilitetsproblemer, har vi likevel håndtert lignende utfordringer som har krevd raske og effektive løsninger.

For eksempel opplevde et teammedlem betydelige forsinkelser på grunn av ekstreme værforhold tidlig i januar, noe som forårsaket en forsinkelse i vedkommende ankomst til et kritisk møte. Teamet responderte ved å raskt omstrukturere dagsordenen for å utnytte tilgjengelig tid effektivt. I februar ble et annet teammedlem syk, noe som nødvendiggjorde en omfordeling av oppgaver blant de øvrige teammedlemmene for å opprettholde prosjektets momentum.

Vårt prosjektarbeid ble negativt påvirket som følge av en uventet og plutselig renovering av Sikri-bygget, en situasjon vi ikke hadde inkludert i våre risikoestimater. Dette førte til at vi midlertidig mistet tilgang til vårt vanlige møterom i bygget. I løpet av den to uker lange renoveringsperioden var vi nødt til å tilpasse oss ved å arbeide eksternt. For å takle denne

utfordringen, etablerte vi online tilkoblinger og gjennomførte vårt arbeid digitalt, noe som var vår primære løsning for å minimere forstyrrelser i prosjektarbeidet.

Hver hendelse som har blitt dokumentert i vår risikologg understreker viktigheten av proaktivitet, tilpasningsevne, og evnen til å respondere hurtig. Gjennom en effektiv risikohåndteringsprosess har vi ikke bare klart å jobbe sammen under press, men vi har også utviklet beredskapsplaner som sikrer at vi kan håndtere mulige hindringer på en smidig måte.

Disse opplevelsene har bidratt til å forsterke vår forståelse for betydningen av å forvente og håndtere det uventede, en lærdom som vil være en uvurderlig ressurs i fremtidige prosjekter.

8. Resultater

Dette prosjektet har resultert i en vellykket webapplikasjon, QuickFix, for å effektivisere håndteringen av kundehenvendelser hos Sikri. Underveis i prosjektet ble det gitt kontinuerlig tilbakemelding fra både prosjektgruppen og Sikri. Dette førte til justeringer og forbedringer av applikasjonen for å sikre at den oppfylte de spesifikke behovene til kundeservice. Utfordringer knyttet til prosjektet inkluderte håndtering av endringsforespørsler, prioritering av oppgaver innenfor sprintene, integrasjonsutfordringer, håndtering av omfangsspredning, testutfordringer, ressursbegrensninger og kommunikasjonsproblemer. Agile metodikken med Scrum og tett samarbeid med Sikri bidro til å løse disse utfordringene effektivt.

Selv om applikasjonen ennå ikke er i bruk i den virkelige verden, mangler vi data om resultatene. Imidlertid, basert på innsikten fra både brukerne og produktlederen, har vi kunnet foreta noen estimater om mulige utfall. Disse mulige utfallene indikerer at QuickFix kan oppnå følgende mål når den tas i bruk:

- Redusert saksbehandlingstid for kundeservicemedarbeidere.
- Økt effektivitet i håndteringen av enkle og kategoriserte kundehenvendelser.
- Forbedret kundetilfredshet gjennom raskere løsning av problemer.
- Sikrere håndtering av brukerdata gjennom JSON-filer og kvalitetssikringstiltak.
- Fleksibelt og tilpasningsdyktig system for fremtidige behov.

Det ble også utviklet omfattende dokumentasjon for applikasjonen. Denne dokumentasjonen ble imidlertid fjernet fra rapporten etter forespørsel fra Sikri, da den inneholder sensitiv informasjon som ikke er ment for offentlig deling.

Gjennomføringen av dette prosjektet har gitt oss en rekke innsikter og lærdommer som strekker seg utover de umiddelbare tekniske resultatene. Disse erfaringene har ikke bare forbedret vår forståelse av kompleksiteten i programvareutvikling, men også styrket vår evne til å arbeide effektivt som et team og i samspill med eksterne stakeholdere. Spesielt har vi lært viktigheten av:

- Smidig metodikk og effektiv prosjektstyring.
- Kommunikasjon og samarbeid i teamet.

- Kvalitetssikring og testing for å sikre robust programvare.
- Tilpasning til endringer og håndtering av uforutsette utfordringer

9. Refleksjon og konklusjon

Til tross for dette var det mye mer som gjerne skulle blitt gjort, men begrensede tid og ressurser satte en stopper for det. Derfor bestemte gruppen seg for å fokusere all energi så effektivt som mulig, ved å fordele den mellom forelesninger, leveringer, presentasjoner, oppgaver, rapporter, møter med veilederen og møter med bedriften. Etter prosjektet har alle gruppemedlemmene tilegnet seg betydningsfulle kompetanser og innsikter innen fullstack utvikling, i tillegg til prosjektledelse

Gruppe 15 fikk høre om Sikri sin prosjektoppfriskning for første gang i 2023, og på grunn av både de tekniske utfordringene og selskapets attraktivitet ønsket hele gruppen sterkt å få dette prosjektet. Og vi var heldige nok til å få det. Dette prosjektet, som vi startet fra bunnen av ved å åpne en kodingside, ble virkelig vårt barn gjennom prosjektstyringen vi implementerte. Vi ga det navnet QuickFix. Nå venter vi stolt på den dagen vårt prosjekt, skapt av gruppen, skal brukes i virkeligheten. QuickFix-prosjektet løste et eksisterende problem internt i Sikri-selskapet. Det å ha et eksempel fra virkeligheten gjorde oss enda mer begeistret. Vi trodde at problemene som kundeservice mottok kunne løses uten teknisk kunnskap. Denne løsningen virket mer bærekraftig og effektiv for oss alle. Nå kan en Sikri-kundeservicemedarbeider logge seg inn på QuickFix og løse et ekte teknisk problem fra en ekte kunde uten behov for kodingskunnskaper. Å være en del av denne teknologiske utviklingen, fremskrittet og løsningen er svært stolt for oss.

I prosessen med dette prosjektet møtte vi noen ganger tekniske utfordringer vi ikke visste om, opplevde nedturer og oppturer i motivasjon, og følte styrken ved å jobbe som et team på fem. Vi så fordelene med det vi hadde lært i studiene våre. Denne langsiktige og utfordrende prosessen lærte oss alle mye. Vi er veldig glade for å ha levert dette prosjektet med suksess.

10. Litteraturliste

Abras, C., Maloney-Krichmar, D., & Preece, J. (u.å.). User-Centered Design. Hentet fra https://shorturl.at/bpV28

Abramov, D., & Nabors, R. (2023, Mart 16). Introducing react.dev. Hentet fra https://react.dev/blog/2023/03/16/introducing-react-dev

Ahmed, B., Kaur, A., 2022. "Effectiveness of Automated Testing in Agile Environments." Journal of Software Engineering and Applications, 35(3), pp. 145-159.

Azure. (u.å.). DevOps at Microsoft. Hentet fra https://azure.microsoft.com/nb-no/solutions/devops/devops-at-microsoft/

Benyon, D. (2019). Designing User Experience: A guide to HCI, UX and interaction design (4th edition)

Božić, Velibor. (2023 Februar). DevOps - Software Development Methodology. DOI: 10.13140/RG.2.2.11626.80327

Cohn, M. (2006). Agile Estimating and Planning. Prentice Hall

Dam, R. F. and Teo, Y. S. (2024, February 8). Personas – A Simple Introduction. Interaction Design Foundation - IxDF. Hentet fra https://www.interaction-design.org/literature/article/personas-why-and-how-you-should-use-them

Dimitrijević, S., Jovanović, J., Devedžić, V. (2015). A comparative study of software tools for user story management . Information and Software Technology. 57: 352-368. DOI:10.1016/j.infsof.2014.05.012

Fredriksen, L. E., Intaraphasuk, K., Hedemark, J., Erdvik, M., Nergaard, E. D., & Ramsdal, M. (2021). Smidig utvikling av utlånssystem (s. 50)

Gallis A.S. (2023). proof of concept (POC). Hentet fra https://www.techtarget.com/searchcio/definition/proof-of-concept-POC

Grenning, J. (2002). Planning Poker or How to avoid analysis paralysis while release planning. Hentet fra http://renaissancesoftware.net/files/articles/PlanningPoker-v1.1.pdf

Highsmith, J. (2002). Agile software development ecosystems

Hoel, I. (2006). *Flate organisasjoner: Studie av en to-nivå modell i en kommune.* Hentet fra: https://openaccess.nhh.no/nhh-xmlui/bitstream/handle/11250/167951/Hoel%20Ida%202006.pdf?sequence=1

Hudda, S., Mahajan, R., & Chopra, S. (2016). Prioritization of User-Stories in Agile Environment. Indian Journal of Science and Technology, 9(45). DOI: 10.17485/ijst/2016/v9i45/105069

Javapoint., (u.å). Software Documantation. Javapoint.com. Hentet fra https://www.javatpoint.com/software-documentation

Kaur, J., Singh, O., Anand, A., & Agarwal, M. (2023). A goal programming approach for agile-based software development resource allocation. Decision Analytics, 6, 100146. https://doi.org/10.1016/j.dajour.2022.100146

Kniberg, H., & Skarin, M. (2010). Kanban and Scrum - Making the Most of Both. Hentet fra https://www.infoq.com/minibooks/kanban-scrum-minibook/

Larson, E. W., & Gray, C. F. (2021). Prosjektledelse: Den Managerial Process (8. utg.). McGraw-Hill Education.

Lewis, W.E., 2019. Software Testing. 2nd ed. New York: CRC Press.
Patton, R., 2020. Software Testing. Sams Publishing.

Lucid Content. (u.å.). Using the Fibonacci scale in Agile estimation. LucidChart. https://www.lucidchart.com/blog/fibonacci-scale-for-agile-estimation

Microsoft Learn. (2022). What is version control? Hentet fra https://learn.microsoft.com/en-us/devops/develop/git/what-is-version-control

Microsoft Learn. (2022). Create web APIs with ASP.NET Core. Hentet fra https://learn.microsoft.com/en-us/aspnet/core/web-api/?view=aspnetcore-6.0

Microsoft Learn (2022). Routing to controller actions in ASP.NET Core. Hentet fra https://learn.microsoft.com/en-us/aspnet/core/fundamentals/routing?view=aspnetcore-6.0

Microsoft Azure (u.d). Azure benefits and incentives. Hentet fra https://azure.microsoft.com/en-us/pricing/offers

Moe, N.B., Dingsøyr, T., & Dybå, T. (2008). Understanding Self-organizing Teams in Agile Software Development. Hentet fra https://www.researchgate.net/publication/220518111

MUI. (u.å.). React-komponenter for raskere og enklere webutvikling. Hentet fra https://mui.com/

Nielsen Norman Group. (2018, 25. Februar). UX Expert Reviews. Hentet fra https://www.nngroup.com/articles/ux-expert-reviews/

Nyvoll C. J. (2023, 23. januar). 7 steg mot kundedrevne prosesser. Hentet fra https://www.kristiania.no/kunnskap-kristiania/2023/01/7-steg-mot-kundedrevne-prosesser/

Radigan, D. (n.d.). Product backlog - What is it & how to create one. Atlassian. Hentet fra https://www.atlassian.com/agile/scrum/backlogs

Raeburn, A. (2024). What is a product backlog? (And how to create one), hentet fra: https://asana.com/id/resources/product-backlog

Ramsøy, C. (2022). En kort introduksjon til Scrum, hentet fra: https://www.visma.no/blogg/en-kort-introduksjon-til-scrum/

React. (u.å.). En deklarativ, effektiv, og fleksibel JavaScript-bibliotek for å bygge brukergrensesnitt. Hentet fra https://reactjs.org/

React Bootstrap. (u.å.). Det mest populære front-end rammeverket gjenoppbygget for React. Hentet fra https://react-bootstrap.github.io/

React Toastify. (u.å.). Gjør varsler til en enkel og morsom oppgave. Hentet fra https://fkhadra.github.io/react-toastify/

Rehkopf, M., (2024.). Jira Burndown chart tutorial. https://www.atlassian.com/agile/tutorials/burndown-charts

Roth, D., Anderson, R., & Luttin, S. (2023, Nisan 10). Overview of ASP.NET Core. Hentet fra https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-8.0

Rubin, K.S. (2012). Essential Scrum: A Practical Guide to the Most Popular Agile Process. Addison-Wesley Professional.

Salian, V. (2020). Product quality vs Code quality. Hentet fra https://vijeshsalian.medium.com/product-quality-vs-code-quality-3c7a7a662656

Scrum Guides. (2020). Scrum Guide. Hentet fra: https://scrumguides.org/scrum-guide.html

Sikri (u.å.). Hentet fra: https://www.sikri.no/om-sikri

SpecFlow. (2022). "SpecFlow official documentation". Hentet Fra https://specflow.org/documentation/

Stetsenko, R. (u.å.). https://www.toptal.com/app/pixel-perfect-ios-ui-design

Sutherland, J., & Schwaber, K. (2020). The Scrum Guide. Hentet fra https://www.scrumguides.org/scrum-guide.html

Ux design institue. (2022). "The Value of UX Design." Hentet fra: https://www.uxdesigninstitute.com/blog/the-value-of-ux-design/

Xunit. (2022). "Xunit documentation". Hentet Fra https://xunit.net/docs/getting-started/netcore/cmdline

11. Appendiks

11.1 Appendiks 1 – Uttalelse fra Sikri

14.05.2024 Kristiansand

Uttalelse fra Sikri

Bachelorgruppen og Sikri ble enig om samarbeid november 2023. Gruppen ble gitt i oppgave å gjøre POC av en løsning for å forenkle arbeidet til kundesenteret og drift i Sikri gjennom å tilgjengeliggjøre komplisert konfigurasjon på ny lettere måte.

Under arbeidet med design ble det gjennomført intervju med brukere og brukertesting av design før utvikling og tilvarende gjentatt når utvikling nær ferdig. Brukerhistorier ble definert, og samtlige som var strengt nødvendige ble gjennomført.

Bachelorgruppen og representant fra Sikri hadde regelmessige møter og kommuniserte digitalt gjennom Slack. Kommunikasjonen fungerte godt, med klare kommunikasjonsmuligheter. Bachelorgruppen var klar på behov og samarbeidet bra med oss i Sikri. Gruppe fremstod profesjonelle, delte bra på ansvaret og virket å ha et godt samarbeid internt.

POC-en går inn som et viktig bidrag i arbeidet hos Sikri med å forenkle hvordan vi jobber med konfigurasjon, og vil bli vurdert satt ut i drift som den er eller videreutviklet etter behov. Gruppen stilte uoppfordret med grundig teknisk dokumentasjon som gjør eventuell bruk og videreutvikling enkelt for oss i Sikri.

Med vennlig hilsen

Kristoffer Stokkeland, Tech-Lead Elements Møte
Sikri AS

Sikri AS • Dronning Mauds gate 10, 0250 Oslo
905 30 049 • post@sikri.no • www.sikri.no
Org.nr.: 922 308 055

11.2 Appendiks 2 – Egenvurdering

Umit Yildirim

I løpet av dette prosjektet tok jeg på meg ansvaret som gruppeleder, med ansvar for å koordinere den generelle organisasjonen. Jeg var ansvarlig for å fastsette møtedager og -tider, overvåke møter og sørge for at oppgaver ble tildelt og utført til rett tid og på riktig måte. Jeg spilte også en nøkkelrolle i organiseringen av arbeidssteder. Selv om jeg deltok i alle faser av prosjektet, tok jeg spesielt på meg en viktig rolle i utviklingen av frontend. I samarbeid med Orhan og Buse, fullførte vi suksessfullt en stor del av frontend-seksjonen. Jeg implementerte også prosjektet for å fungere med Docker, en teknisk oppgave som var avgjørende for prosjektets suksess.I sprint 5 fungerte jeg som master, ansvarlig for distribusjon og oppfølging av oppgaver. I tillegg hadde jeg en aktiv rolle i å distribuere oppgaver, gjøre justeringer og bidra til skrivingen av prosjektet. Erfaringene jeg har fått gjennom dette prosjektet og de ansvarsrollene jeg har tatt, har tillatt meg å forbedre både mine tekniske ferdigheter og ledelsesevner. Mine evner til å samarbeide og koordinere innad i teamet spilte en kritisk rolle i å fullføre prosjektet med suksess.

Orhan Yildirim

I dette prosjektet bidro jeg betydelig til forståelsen av problemdomene og mulige løsninger. Jeg spilte en aktiv rolle i utforming av prosjektarkitekturen. Spesielt i prosjektets frontend-del, designet jeg arkitekturen for innsamling av data fra relevante seksjoner og sendte dem i riktig format til backend, som forventet av backend. Jeg implementerte også begrensninger for å forhindre mulige feil inndata fra brukerne. I denne prosessen samarbeidet jeg med Ümit og Buse og bidro betydelig til størstedelen av frontend-seksjonen. Jeg spilte også en aktiv rolle i gjennomføringen av brukertester for prosjektet. I tillegg bidro jeg til skrivingen av rapporten, gjennomgikk rapporten generelt, og identifiserte og rettet mangler. I den sjette sprinten fungerte jeg som Scrum master, ansvarlig for tildeling og oppfølging av oppgaver. Jeg tok på meg aktive roller som å følge oppgaver, gjøre justeringer og bidra til prosjektskrivingen, noe som forbedret både mine ferdigheter og lederegenskaper. Min evne til å samarbeide harmonisk og mine ferdigheter spilte en kritisk rolle i å fullføre prosjektet med suksess.

Samet Demirezen

Selv om mitt hovedansvar gjennom prosjektet var design og utvikling av backend, var det også perioder jeg jobbet med frontend. Jeg spilte en viktig rolle i opprettelsen av den grunnleggende prosjektstrukturen, bestemmelse av filstrukturen og utviklingsplanleggingen. Jeg bidro til utviklingen av klasser og metoder i Controller og tilhørende Services-mappen. Jeg jobbet på flere

måter både i backend og frontend, blant annet ved å sette opp grunnleggende frontend-backend integrasjon. Jeg samarbeidet med mange gruppekamerater både i rapportering og utvikling. Jeg deltok i våre sprintmøter og daglige vurderingsmøter, hvor jeg hadde en aktiv rolle. I en sprint fungerte jeg som scrum master, og sørget effektivt for at oppgavefordelingen og fremdriften gikk jevnt. Til slutt skrev jeg dokumentasjonen for appen for å lette bruk og videreutvikling.

Burak Seymen

I løpet av prosjektet var mitt hovedansvar å arbeide med backend, spesielt å utvikle kontrollere som kommuniserer med Azure. Jeg bidro også til å implementere grunnleggende tester for både backend og frontend, noe som sikret at systemets funksjoner var pålitelige og effektive. Som Scrum Master i de to første sprintene ledet jeg teamet gjennom planleggings- og gjennomføringsfasene, og etter dette tok vi beslutningen om å rotere denne rollen. Dette ga meg verdifull erfaring med både ledelse og samarbeid. I tillegg arbeidet jeg aktivt med rapporten, tilpasset behovene og tilgjengeligheten i prosjektet. Min innsats i disse områdene har ikke bare styrket mine tekniske ferdigheter, men også min evne til å arbeide effektivt i et team.

Busenur Yilmaz

I prosjektet har min hovedrolle vært å håndtere kommunikasjonen mellom ulike parter, inkludert bedrifter, veiledere og fagansvarlige lærere. Jeg tok initiativ til å planlegge intervjuer og tester med andre som bruker UX-design og ansatte hos Sikri. I tillegg til dette, som alle de andre medlemmene i gruppen, var jeg også scrum master i en sprint.

Videre tok jeg også initiativ til å designe prototyper. I tillegg til dette, bidro jeg til frontend-utviklingen sammen med Orhan og Umit, hvor vi samarbeidet om ulike oppgaver. Jeg jobbet spesielt med komponenter og brukervennlige grensesnitt, og løste alle utfordringer som frontend-teamet møtte basert på tilbakemeldinger fra brukerne.

Jeg søkte også kunnskap om GET- og POST-metoder fra Burak, som hjalp meg med å få en bedre forståelse av fullstack-utvikling. Jeg føler meg utrolig heldig for å ha vært en del av både gruppen og prosjektet, da det har gitt meg muligheten til å utfolde mine evner innen kreativitet, sosiale ferdigheter, teamarbeid og utvikling.

11.3 Appendiks 3 – Risikoanalyse

Risiko	Sannsynlighet (1-5)	Konsekvens (1-5)	Totalt risikonivå	Forebyggende tiltak	Reaktive tiltak
Generell sykdom	2	5	10	Spis sunt, sov nok	Gå gjennom møtenotater nøye
Datatap	1	5	5	Automatisk lagring og versjonskontroll	Ta backup, omstart om nødvendig
Manglende deltagelse	4	2	8	Ikke tolerer forsinkelser, bruk påminnelsesalarmer	Gi bot i form av å spandere pizza
Lett sykdom	2	3	6	Spis sunt, sov nok	Arbeid hjemmefra
Mangel på motivasjon	3	1	3	Øk motivasjon gjennom sosiale aktiviteter	Organiser teamaktiviteter som en tur til Berlin
Dårlig moral i gruppen	3	3	9	Oppmuntre hverandre, snakk sammen	Finn løsninger før neste møte
Utilstrekkelig kommunikasjon	2	2	4	Skriv møtereferater/daglig scrum	Løs problemer gjennom diskusjon og opprettholdkommunikasjon
Manglende kompetanse	4	2	8	Rådfør deg med erfarne, deleger oppgaver	Be om hjelp eller søk informasjon
Internettproblemer	1	5	5	Utenfor vår kontroll	Fokus på andre oppgaver som ikke krever internett
Dårlige kode standarder/dokumentasjon	4	4	16	Bruk selskapets dokumentasjonsstandarder	Skriv kode basert på dokumentasjon
Generelt fravær	4	3	12	Ikke gå glipp av møter	Gi forhåndsvarsel
Konflikter og uenigheter i gruppen	1	3	3	Fremme en kultur for å akseptere ulike meninger	Diskuter for å løse uenigheter
Gruppemedlemmer får ikke nok søvn	4	2	8	Sørg for personlig ansvar for tilstrekkelig søvn	Tilrettelegg for spesielle behov
Gruppemedlemmer jobber for mye	2	3	6	Fordel arbeidsoppgaver jevnt	Sørg for at alle bidrar og har det bra
Feil estimering av sprint	4	3	12	Planlegge med ekstra tid	Fullfør manglende oppgaver i neste sprint

Svært lav	Lav	Middels	Høy	Veldig høy	Kritisk

11.4 Appendiks 4 – Risikomatrise

	Konsekvens					
Sannsynlighet		Ubetydelig (1)	Liten (2)	Moderat (3)	Stor (4)	Katastrofal (5)
	Veldig sannsynlig (5)					
	Sannsynlig (4)		Manglende deltagelse (8) Manglende kompetanse (8) Gruppemedlemmer får ikke nok søvn (8)	Generelt fravær (12) Feil estimering av sprint (12)	Dårlige kode standarder /Dokumentasjon (16)	
	Mulig (3)	Mangel på motivasjon (3)		Dårlig moral i gruppen (9)		
	Usannsynlig (2)		Utilstrekkelig kommunikasjon (4)	Lett sykdom (6) Gruppemedlemmer jobber for mye (6)		Generell sykdom (10)
	Veldig usannsynlig (1)			Konflikter og uenigheter i gruppen (3)		Datatap (5) Internett problemer (5)
Svært lav	Lav	Middels	Høy	Veldig høy	Kritisk	

11.5 Appendiks 5 – Risikologg

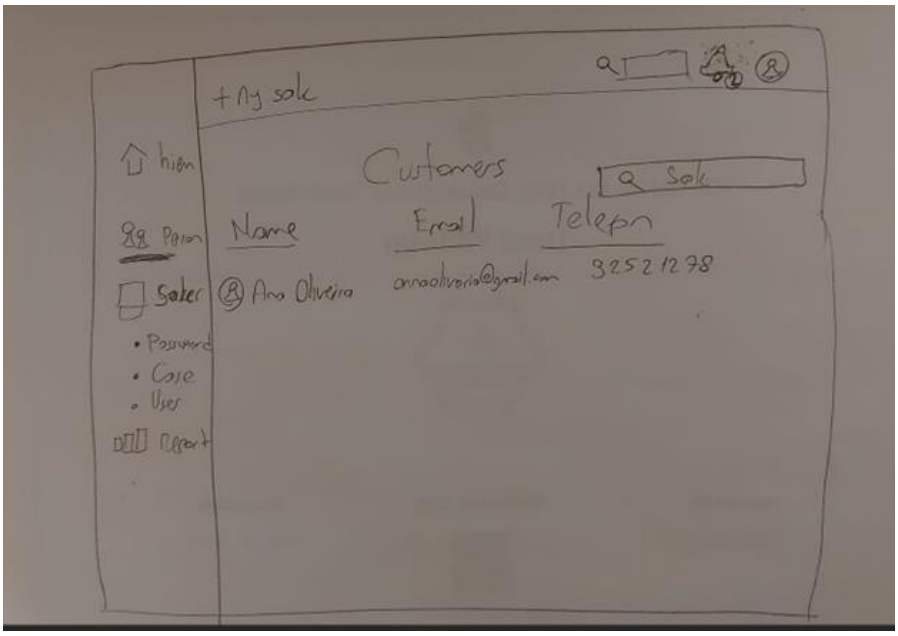
Nummer	Risiko Nummer	Risikotype	Startdato	Sluttdato	Kommentar
1	8	Ekstremt vær	15.01.24	15.01.24	En uventet snøstorm forsinket Ümit Yildirim under hans reise fra Grimstad til Kristiansand. Han ble sittende fast på en buss som var forsinket på grunn av et trafikkuhell.
2	8	Ekstremt vær	20.01.24	20.01.24	Buse Yilmaz ankom sent til et møte etter et betydelig snøfall som hindret henne i å få bilen ut av parkeringsplassen.
3	10	Sykdom	15.02.24	22.02.24	Orhan Yildirim utviklet influensasymptomer og kunne ikke delta i møtene, noe som krevde at resten av teamet måtte omdisponere hans oppgaver.
4	12	En annet Forlesning Forpliktelser	15.03.24	15.03.24	På grunn av akademiske forpliktelser til en annet forlesning han tok, måtte Samet Demirezen bruke tid på lekser og skolegang, noe som førte til at han gikk glipp av noen planlagte møter.
5	4	Personlige forpliktelser	20.03.24	27.03.24	Samet Demirezen hadde sitt bryllup og påfølgende forberedelser, noe som resulterte i hans fravær fra to planlagte møter.
6	5	Manglende infrastruktur	05.04.24	22.04.24	Mangel på tilstrekkelig utstyr på Sikri-kontoret forhindret teamet i å utføre nødvendige tester, noe som førte til en ukes forsinkelse.

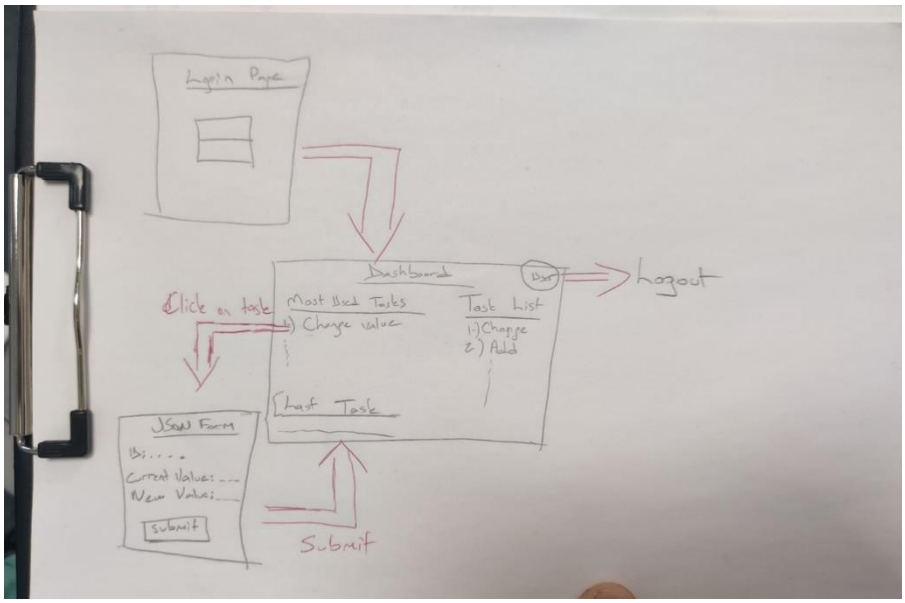
11.6 Appendiks 6 – Bruker historier med MoSCoW

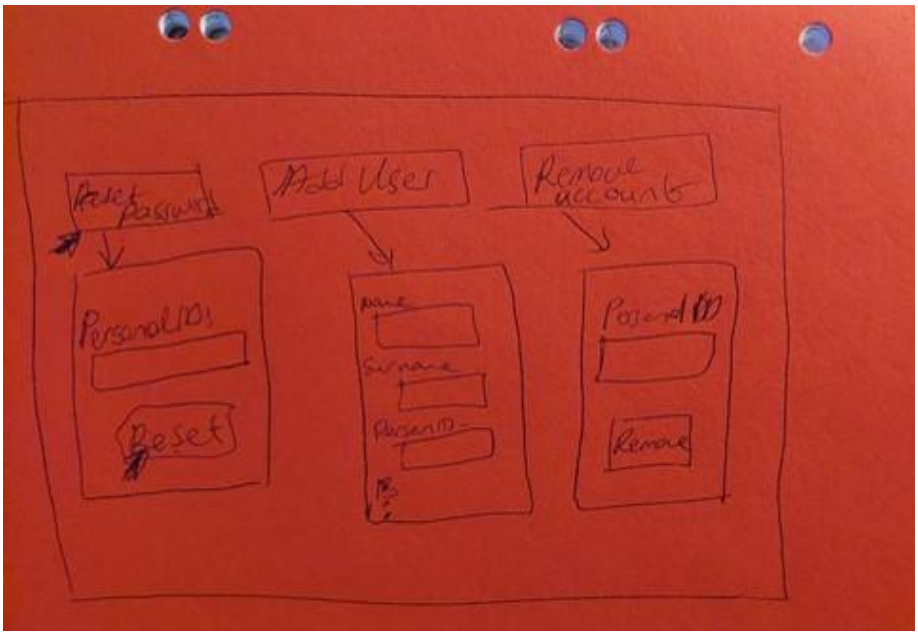
ID	Prioritering	Brukerhistorie	Argument
1	Must Have	Som en bruker ønsker jeg å kunne oppdatere innstillinger for en kunde gjennom json-filer.	Denne funksjonen er prioritert som «Must Have» fordi dette vår siste målet av hele prosjektet. Dette er essensielt for effektiv håndtering av saker og kundeinformasjon.
2	Must Have	Som en bruker ønsker jeg en brukervennlig, minimalist grensesnitt og enkel navigasjon for å forbedre min generelle opplevelse med applikasjonen.	Denne funksjonen rangeres som et «Must Have» fordi den forbedrer applikasjonsopplevelsen, selv for brukere uten teknisk kunnskap.
4	Must Have	Som en bruker ønsker jeg å kunne logge meg inn i applikasjonen for å sikre at bare autoriserte brukere har tilgang til systemet.	Funksjonen er et Must Have, fordi dette systemet kun skal kunne benyttes av kundesenter hos Sikri.
5	Must Have	Som en bruker, ønsker jeg å kunne finne innstillinger i kategorier, for å forenkle prosessen og bidra til tidsbesparelser.	Denne funksjonen er et Must Have, fordi, dette systemet inneholder sensitiv og kritisk data som ikke må endres feilaktig.
6	Must Have	Som en bruker, ønsker jeg at systemet viser detaljer om endringer på GitHub, for å oppnå bedre sporbarhet og forståelse av implementerte endringer.	Dette kravet er klassifisert som «Must Have» fordi det sikrer at systemet tilbyr en tilpasset og inkluderende brukeropplevelse ved å møte individuelle språkbehov, noe som er essensielt for brukertilfredshet og tilgjengelighet.
7	Should have	Som en bruker ønsker jeg å kunne tilpasse grensesnittets farger og temaer, slik at jeg kan tilpasse opplevelsen min etter mine preferanser.	Funksjonen klassifiseres som «Should Have» fordi den gjør systemet mer brukervennlig for bruker, men det er ikke en kritisk funksjon for at systemet skal fungere.
8	Should Have	Som en bruker ønsker jeg å kunne velge språk i systemet for å tilpasse	Dette kravet er klassifisert som «Should Have» fordi det forbedrer systemets tilgjengelighet og brukertilfredshet ved å tilby en tilpasset og inkluderende

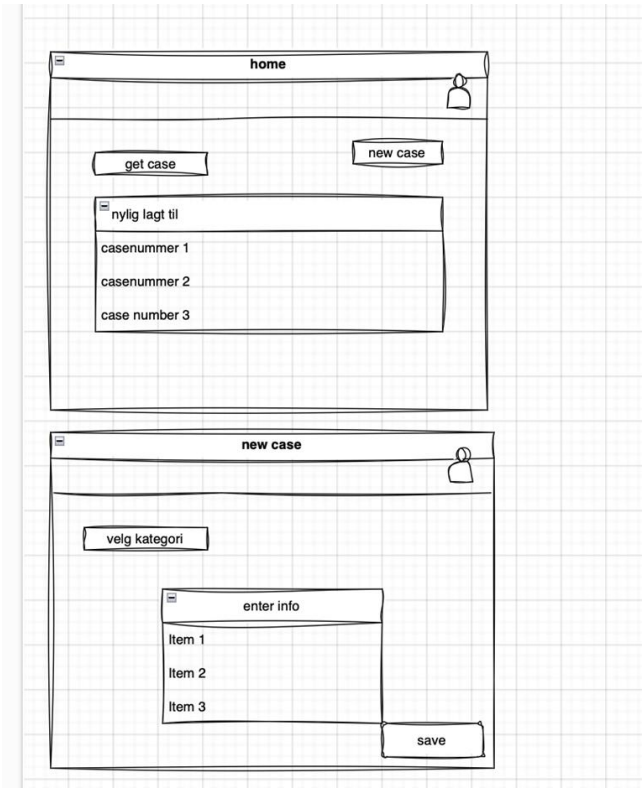
		opplevelsen til mine preferanser og behov.	brukeropplevelse som møter individuelle språkbehov.
9	Should have	Som en bruker, ønsker jeg å kunne se en historikk over de siste endringene, for å ha tilgang til relevant informasjon om systemets endringer.	Denne funksjonen er rangert som "Should have" fordi den, selv om den kunne være ønskelig, ikke er høyt prioritert grunnet tidsbegrensninger i prosjektet.
10	Could have	Som en bruker, ønsker jeg å bruke QuickFix og Zendesk samtidig, for å forbedre effektivitet.	Denne funksjonen er klassifisert som "Could have" fordi den potensielt kan forbedre brukeropplevelsen, men den ligger utenfor prosjektets omfang eller produkteierens forventninger.
11	Won’t have	Som en bruker, ønsker jeg begrensninger for innlogging fra flere enheter samtidig, for å oppnå at applikasjonens mål ikke inkluderer mobilbruk.	Denne funksjonen er vurdert som "Won’t have" fordi målet med applikasjonen passer ikke for mobilbruk.

11.7 Appendiks 7 – Skisser






Reset password
Add User
Remove account
PersonalID:
Reset
Name
Surname
PersonID
Remove


home
get case
new case
nylig lagt til
casenummer 1
casenummer 2
case number 3
new case
velg kategori
enter info
Item 1
Item 2
Item 3
save

11.8 Appendiks 8 –Intervju spørsmål

DOKUMENT FOR INTERVJU OM PROBLEMOMRÅDET

Prosjektbeskrivelse:

Prosjektet fokuserer på å optimalisere brukeropplevelsen for en kompleks organisatorisk plattform. Målet er å digitalisere kundesenter oppgaver. Vårt applikasjonen sikrer at kundesenter kan utføre oppgavene sine effektivt og intuitivt.

Intervjuguide:

Vi ønsker å gjennomføre et intervju med dere som SIKRI-ansatte for å få en dypere forståelse av problemområdet. Vi har 6 åpne spørsmål, og dere kan kommentere alt mens vi snakker sammen. Siden det er åpne spørsmål, kan jeg bruke svarene deres og også stille oppfølgingsspørsmål utenom de spørsmålene som allerede er satt opp. Takk for at dere deltar i intervjuet.

1. **Vanlige Saker i Kundeservice:**

"Hvilke typer saker møter kundeserviceteamet mest? For eksempel, vi vet at passordendringer, endringer av saksnummer eller oppdateringer av brukernavn er vanlige. Er det andre hyppige problemer basert på statistiske data?"

2. **Prioritering av Problemløsning i Appen:**

"Blant disse problemene, hvilke ønsker du å prioritere for løsning i vår app? Og hvorfor?"

3. **Prosess for Håndtering av Saker:**

"Kan du beskrive den nøyaktige prosessen fra en sak åpnes til problemet er løst og tilbakemelding er gitt til kunden? Vi er spesielt interessert i å forstå trinnene og rollene involvert i hver fase."

4. **Kriterier for Å Sende Saker til OPS:**

"Hvilke typer saker mener du bør eskaleres til OPS, og hvilke kan potensielt løses gjennom en nettbasert applikasjon? Hvilke kriterier bør brukes for å ta denne avgjørelsen?"

5. **Identifisering av Enkle Saker for In-App Løsning:**

"Fra ditt perspektiv, hvilke typer saker kan klassifiseres som 'enkle' og håndteres direkte i appen, uten behov for å eskalere dem til OPS?"

6. **Kunde Tilbakemeldingsprosess:**

"Hvordan fungerer prosessen med kundetilbakemelding etter at en sak er løst? Hvilke metoder brukes for å måle kundetilfredshet?"

11.9 Appendiks 9 – Gruppekontrakt

Avtale om gjennomføring av IS-304 Bacheloroppgave i IT og informasjonssystemer

Formålet med avtalen er å sikre gjensidig utbytte av samarbeidet, både for virksomheten og studenten. Avtalen inngås når studentene har fått godkjent skisse til tema og arbeidsmåte for bacheloroppgaven.

Virksomheten ansvar

- Virksomheten oppnevner en kontaktperson.
- Virksomheten fastsetter bacheloroppgaven i samarbeid med studenten.
- Virksomheten skal orientere studenten om de regler og retningslinjer som gjelder for virksomhetens ansatte, herunder eventuelt regelverk knyttet til personvern og taushetsplikt, og gi studenten tilgang til nødvendige systemer.
- Virksomheten skal så raskt som mulig ta opp eventuelle problemer i forbindelse med samarbeidet med studenten og universitetets representant.

Kontaktperson

Kristoffer Stokkeland, kristoffer.stokkeland@sikri.no, 48215910

Universitetets ansvar

- Universitetet oppnevner en representant (normalt veileder eventuelt emneansvarlig) som virksomhet og student kan forholde seg til i forbindelse med gjennomføring av samarbeidet.
- Universitetet forbereder studenten på samarbeidet og de kriterier som gjelder for gjennomføring.
- Representant fra universitetet følger opp virksomheten og studenten under prosjektperioden.
- Representant fra universitetet skal følge opp eventuelle meldinger om problemer i forbindelse med samarbeidet fra student eller virksomhet.

Universitetets representant (Veileder)

Studentens ansvar

- Studenten skal møte til avtalt tid i virksomheten.
- Dersom studenten pga. sykdom eller annen årsak ikke kan møte som avtalt, skal studenten gi beskjed til virksomheten så snart som mulig.
- Studenten forplikter seg til å følge de regler og retningslinjer som gjelder for virksomhetens ansatte.
- Studenten skal så raskt som mulig ta opp eventuelle problemer i forbindelse med samarbeidet med virksomhet og universitetets representant.
- Studenten skal ikke motta lønn for arbeidet.
- Studenten dekker reisekostnader til/fra virksomheten (og eventuelle oppholdskostnader) selv med mindre annet er avtalt med universitetet.

Rettigheter

Dersom annet ikke er avtalt så er det virksomheten som har alle rettigheter til produktet.

Tillegg til avtale.

Annet som avtalepartnere har blitt enige om, fyll inn (for eksempel hvis virksomheten ønsker at arbeidet inkludert bacheloroppgaven skal holdes konfidensiell).

Sted/dato

Signatur:

Kristoffer Stokkeland

Kristoffer Stokkeland

For virksomhet

Student

Orhan Yildirim

Burak SEYMEN

Ümit Yildirim

Buse Yilmaz

For Universitetet i Agder, Emneansvarlig, Hallgeir Nilsen, hallgeir.nilsen@uia.no 41565684

Hallgeir Nilsen

11.10 Appendiks 10 – Eksperttest intervjuguide

INTERVJUDOKUMENT FOR EXPERT UX DESIGN

Prosjektbeskrivelse:

Prosjektet vårt fokuserer på å optimalisere brukeropplevelsen (UX) for en kompleks organisatorisk plattform. Målet er å forbedre navigasjon, funksjonalitet og brukervennlighet for å sikre at brukerne kan utføre oppgavene sine effektivt og intuitivt.

Intervjuguide:

Vi ønsker å intervjue erfarne UX-designere som har betydelig erfaring med å utvikle brukergrensesnitt for komplekse systemer. Formålet med intervjuet er å få innsikt i deres tanker og tilnærminger til å optimalisere brukeropplevelsen. Under intervjuet vil jeg ta notater mens du deler dine meninger og observasjoner.

Vi vil starte med noen konkrete oppgaver hvor jeg vil be deg om å utføre handlinger som "*kan du trykke der*". Deretter vil vi gå dypere inn i spørsmål om designprinsipper og generelle forbedringer i systemet.

Vi ønsker å forstå din tilnærming til UX design, samt dine tanker om hvordan vi kan forbedre vårt system for å møte brukernes behov og forventninger på best mulig måte.

Intervjuspørsmål:

Følge-instruksjoner
1. Be personen om å logge inn i systemet med brukernavn og passord. 2. Finn en bestemt organisasjon med organisasjonsnummeret "xxxxxxx": 3. Endre date interval innstillingene for denne organisasjonen 4. Finn en annen bestemt organisasjon med organisasjonsnummeret "xxxx": 5. Endre publikasjonens fil for denne organisasjonen 6. Få oversikt over den siste oppgaven (task) 7. Be personen om å laste ned den nødvendige informasjonen fra systemet og eksportere den til en PDF-fil. 8. Be personen om å endre språkinnstillingene i systemet til ønsket språk. 9. Avslutt økten ved å logge ut av systemet.

UX-aspekter:

1. Hvordan kan vi sikre vi beste måte at constraint-prinsippet blir implementert i applikasjonen slik at brukeren unngår feil?
2. Kan du dele dine tanker om responsivt design og hvordan du ville sikre at grensesnittet fungerer sømløst på forskjellige enheter og skjermstørrelser?
3. Hvordan vurderer du betydningen av tilgjengelighet i UX-design, og hvordan ville du sikre at plattformen oppfyller tilgjengelighetsstandarder for alle brukere?
4. Kan du diskutere din tilnærming til brukerundersøkelser og testing i UX-designprosessen, og hvordan du ville integrere disse metodene for å validere designvalg og identifisere forbedringsområder?
5. Er det noen spesifikke utfordringer eller positive erfaringer du vil fremheve i forbindelse med bruk av systemet for endring filer.

11.11 Appendiks 11 – Brukertest intervjuguide

INTERVJUDOKUMENT FOR BRUKERTEST

Prosjektbeskrivelse:

Prosjektet fokuserer på å optimalisere brukeropplevelsen (UX) for en kompleks organisatorisk plattform. Målet er å forbedre navigasjon, funksjonalitet og brukervennlighet for å sikre at brukerne kan utføre oppgavene sine effektivt og intuitivt.

Intervjuguide:

Vi ønsker å gjennomføre brukertester med faktiske brukere av systemet for å samle inn data om brukervennlighet, funksjonalitet og tilfredshet. Brukertestene vil hjelpe oss med å identifisere problemer som ikke nødvendigvis blir oppdaget gjennom interne tester. Vi vil undersøke hvordan faktiske brukere interagerer med systemet og hvordan de opplever forskjellige aspekter av brukergrensesnittet og navigasjonen.

Intervjuspørsmål:

Følge-instruksjoner
1. Logg inn i systemet med brukernavn og passord. 2. Finn en spesifikk organisasjon ved å bruke organisasjonsnummeret "xxxxxxx". 3. Endre datointervallinnstillingene for denne organisasjonen. 4. Finn en annen spesifikk organisasjon med organisasjonsnummeret "xxxx". 5. Endre filen for publikasjonen for denne organisasjonen. 6. Få en oversikt over den siste oppgaven (task). 7. Last ned nødvendig informasjon fra systemet og eksporter dette til en PDF-fil. 8. Endre språkinnstillingene i systemet til ønsket språk. 9. Avslutt økten ved å logge ut av systemet.

UX-aspekter:
1. Hvilket førsteinntrykk hadde du når du først landet på appens påloggingsside? Var designet og layouten inviterende og intuitiv? 2. Var påloggingsknappen lett å finne og tydelig synlig? Beskriv din opplevelse med å finne og bruke den. 3. Da du valgte en oppgave på dashbordet, var instruksjonene og nødvendige feltene tydelige og lett tilgjengelige? 4. Beskriv hvordan du opplevde å søke etter og finne spesifikk organisasjonsinformasjon. Var informasjonen nøyaktig og lett tilgjengelig? 5. Opplevde du noen tekniske problemer eller feil mens du brukte appen, spesielt under oppgaveløsning eller logging av aktiviteter? 6. Hvor enkelt var det å endre språkinnstillingene, og ble endringen korrekt reflektert i hele appgrensesnittet? 7. Hvordan vurderer du den generelle brukervennligheten til appen? Følte du at navigasjonen mellom forskjellige seksjoner var logisk og effektiv?

11.12 Appendiks 12 – Prototype lenke

https://www.figma.com/file/FNIZZMk1avwx40Ol1utiPu/QuickFixApp?type=design&node-id=90-749&mode=design&t=PXF8uFahJhsFomba-0

11.13 Appendiks 13 – Dokumentasjon

Dokumentasjon for kjøring og videreutvikling av QuickFix

Gruppe 15

Busenur Yilmaz

Burak Seymen

Orhan Yildirim

Umit Yildirim

Samet Demirezen

Mai 2024

Innholdsfortegnelse for dokumentasjon

! Viktig notat:

Dokumentasjonen er ikke inkludert i rapporten etter forespørsel fra Sikri, da den inneholder sensitiv bedriftsinformasjon.