Norconsult
Informasjonssystemer

Forprosjektrapport

Bachelorprosjekt i Anvendt Datateknologi 2021

Context-sensitive Help

OSLOMET

Gruppe 8
Jostein J. Hauge
Zakariya A. Ibrahim
Elin H. Hegsvold

Innholdsfortegnelse:

1. Presentasjon

1.1 Gruppemedlemmene

Vi er tre studenter fra OsloMet som er i ferd med å avslutte en bachelorgrad i Anvendt Datateknologi. Vi møttes under fadderuka og har jobbet tett sammen gjennom hele studiet. Dette gjør at vi har de beste forutsetninger for et godt samarbeid gjennom bachelorprosjektet.

Jostein J. Hauge
s333943@oslomet.no
+47 988 59 269

Zakariya A. Ibrahim
s326058@oslomet.no
+47 484 31 387

Elin H. Hegsvold
s333939@oslomet.no
+ 47 974 07 320

1.2 Oppdragsgiver

Norconsult Informasjonssystemer AS (NoIS)
post@nois.no
+47 675 71 500
Kjørboveien 16, 1337 Sandvika

«Norconsult Informasjonssystemer AS (NoIS) utvikler, markedsfører og leverer helhetlige IKT-løsninger for prosjektering, bygging og forvaltning av infrastruktur og eiendom.

Selskapet har også betydelig virksomhet knyttet til leveranse av konsulenttjenester innen IT-prosjektledelse, IT-rådgivning, IT-arkitektur, systemutvikling, systemintegrasjon, RPA, maskinlæring og AI innen alle bransjer. I tillegg har også selskapet dyptgående kompetanse innenfor informasjonssikkerhet og cyber security, med egne konsulenter som jobber dedikert innenfor fagområdet.

NoIS er et heleid datterselskap av Norconsult AS. Norconsult er Norges største og en av Nordens ledende tverrfaglige rådgivere rettet mot samfunnsplanlegging og prosjektering, og har gjennom lang erfaring bygget opp et solid fundament som en toneangivende aktør både nasjonalt og internasjonalt.»[1]

1.2.1 Veileder fra NoIS

Marita Ruud Gundersen
marita.ruud.gundersen@norconsult.com
+47 454 01 553
Gruppeleder
For mer informasjon om veileder fra NoIS:

1.2.2 Kontaktperson fra NoIS

Cecilie Voss
Cecilie.Voss@norconsult.com
+47 980 367 72
Seniorrådgiver
For mer informasjon om kontaktperson fra NoIS:

[1] https://www.nois.no/om-oss/om-nois/

1.3 Veileder fra OsloMet

Andrea Arcuri
Andrea.Arcuri@oslomet.no
+47 67 23 71 90
Professor
For mer informasjon om veileder fra OsloMet:

2. Sammendrag

I løpet av dette prosjektet skal det utvikles en context-sensitive help (heretter CSH) for Oslo REG (Renovasjons- og gjenvinningsetaten) sitt interne system *WebDEB*. CSH er en betegnelse på hjelp som er rettet mot en enkelt tilstand eller funksjon i en programvare. Denne skiller seg fra generell hjelp som er rettet mot programvaren i sin helhet. CSH kan tilbys i mange ulike former. Vår CSH skal være en egen modul som gir brukeren informasjon om den siden hen befinner seg på. Den skal tilby søk etter hjelp og en snarvei til support dersom hjelpen i CSH'en ikke er tilstrekkelig. Det skal være enkelt å konfigurere modulen til andre systemer, og på den måten kunne bli en av NoIS sine hyllevarer.

Forprosjektet innebar research på CSH, utforming, ønsket funksjonalitet og implementering av dette. Det ble gjennomført en rekke møter med NoIS for å få innsikt i WebDEB og hvordan Oslo REG bruker dette systemet, samt oversikt over ulike behov knyttet til CSH'en. På bakgrunn av dette ble det utarbeidet en kravspesifikasjon for funksjonelle og ikke-funksjonelle krav, samt fastsatt enkelte rammebetingelser og akseptansekriterier. Det ble bestemt hvilke teknologier og verktøy som skal brukes for å oppnå de fastsatte målene, og hvilken arbeidsmetodikk som skal følges gjennom utviklingens gang. For å kunne måle fremdriften til prosjektet ble det utarbeidet en fremdriftsplan. På bakgrunn av resultatene fra forprosjektet skal en CSH designes, implementeres og testes.

3. Dagens situasjon

Oslo REG har ansvaret for innsamling og håndtering av avfall i Oslo. Til sammen tømmer de 114.000 beholdere fra 330.000 husholdninger hver uke, i tillegg til at de drifter anlegget for avfallssortering og etterbehandling.[2] For å hjelpe dem med å holde styr på det hele bruker de et fagsystem (*WebDEB).* Systemet ble tatt i bruk i 2002 og man ser nå på muligheten for å modernisere løsningen. I disse dager gjennomføres et forprosjekt for WebDEB 2.0 (arbeidstittel), der fokuset er på brukskvalitet og muligheter for å inkludere hjelpe-funksjoner i større grad. Et nytt og moderne system vil naturligvis være enklere å bruke, men behovet for veiledning vil alltid være tilstede. Etaten har ca. 720 ansatte med mange ulike bakgrunner og teknologiske forutsetninger, noe som gjør at behovet for veiledning i systemet er stort. Systemet som er i drift i dag tilbyr brukeren hjelp og veiledning gjennom en tradisjonell form for brukermanual i PDF-format, men dog uten god søkemulighet. Brukerne benytter seg i liten grad av denne og tar heller kontakt med support, da det som oftest er en raskere løsning på problemet deres. En CSH vil gjøre hjelpen lettere tilgjengelig og enklere å anvende for brukeren, samtidig som den vil avlaste support.

4. Mål og rammebetingelser

4.1 Arbeidsmetodikk

Utviklingen skal i all hovedsak følge det smidige prosess-rammeverket Scrum. En smidig utviklingsprosess tilbyr fleksibilitet i forhold til kravspesifikasjonen. Det er åpent for å tilføye krav etterhvert som de dukker opp, samtidig som at ustabile krav ikke hindrer progresjonen. Gjennom research knyttet til forprosjektet ble det klart at dette prosjektet beveger seg ut på noe ukjent farvann. Behovet for fleksibilitet i utviklingsprosessen utelukket de plandrevne prosessmodellene, og dermed sto det mellom Scrum og Kanban. Det vil bli brukt et Kanban Board under implementeringen for å lettere holde oversikt over oppgavene. Med unntak av dette ble metodikkene til Scrum valgt fremfor Kanban. Timeboxing skaper en rytme i utviklingen og gjør det enklere å planlegge møter og andre aktiviteter med oppdragsgiver, enn om man bruker Kanban sin taskboxing. Videre var det et ønske om å benytte seg av fordelene knyttet til daglige møter og sprint reviews. I tillegg er det planlagt månedlige møter der gruppemedlemmene kan få vist frem hva som har blitt gjort til Oslo REG for tilbakemeldinger.

[2] https://www.oslo.kommune.no/etater-foretak-og-ombud/renovasjons-og-gjenvinningsetaten/

4.1.1 Sprinter

I Scrum er det vanlig at sprintene varer 2-4 uker. Kortere sprinter gjør det enklere å gjøre endringer og oppdage problemer raskere. Det vil også gjøre samarbeidet med oppdragsgiver tettere ettersom man “tvinges” til å møtes oftere. Sprint-lengden falt derfor på 2 uker og med en gjennomføringsfase på 12 uker, vil arbeidet blir fordelt på 6 sprinter.

4.2 Rammebetingelser

NoIS har et ønske om at CSH’en enkelt skal kunne tilpasses og integreres i andre applikasjoner, slik at den kan fungere som et hyllevare system. Det er derfor viktig at man ser etter løsninger som gjør det enklest mulig for NoIS å vedlikeholde, videreutvikle og gjøre tilpasninger til andre applikasjoner. Dette innebærer blant annet at vi bruker de samme verktøy og programmeringsspråk som NoIS bruker i dag. Det er også svært viktig at koden er godt strukturert og dokumentert.

Selv om CSH’en skal fungere som en hyllevare, vil den i denne oppgaven tilpasses OsloREG. Siden WebDEB (OsloREG sitt nåværende system) er utdatert og nytt system ikke er på plass, vil CSH’en implementeres i en dummy-applikasjon. Når det kommer til design, må vi forholde oss til Oslo kommune sine retningslinjer hva gjelder bruk av font, farger og logo.

4.3 Kravspesifikasjon

På bakgrunn av en rekke møter med oppdragsgiver er det i samarbeid med dem utarbeidet en foreløpig kravspesifikasjon. Grunnet noe usikkerhet i forhold til at noe lignende ikke har vært å finne på markedet i dag, vil trolig kravspesifikasjonen måtte endres underveis. Da det er noe vanskelig å estimere hvor lang tid implementasjon av enkelte krav vil ta, er de minst viktige kravene plassert under «Ønskede funksjonelle krav» (4.3.3).

4.3.1 Funksjonelle krav:

CSH’en skal gi brukeren informasjon om innholdet i siden hen befinner seg.
CSH’en skal kunne vise demo-videoer der det er aktuelt.
CSH’en skal forklare ord og uttrykk i parent systemet som er potensielt vanskelig å forstå.
CSH’en skal tilby funksjon for søk etter informasjon om systemet.
CSH’en skal tilby brukeren en snarvei til support dersom hen ikke finner hjelpen hen trenger.
CSH’en skal gi brukeren tilbakemeldinger på ønskede og uønskede hendelser i programmet.
Systemet skal ha et eget brukergrensesnitt for konfigurasjon og publisering av informasjon (CMS).

4.3.2 Ikke-funksjonelle krav:

Vårt nye system med CSH skal føre til et system som er lett å anvende og brukervennlig for brukerne. For å tydeliggjøre forventet resultat av det nye systemet har vi utformet noen essensielle ikke-funksjonelle krav:

CSH'en skal ikke være tilgjengelig hvis du ikke er logget inn i parent systemet.
Søkefunksjonen skal ikke bruke mer enn 5 sekunder på å finne evt. resultat.

- 5 sekunder er i utgangspunktet mye for en søkefunksjon, men her vil den søke gjennom en potensielt lang brukermanual, og man må derfor regne med at det tar noe tid. Vi vil likevel jobbe mot at det tar maksimalt 3 sekunder.

CSH'en skal ikke dekke mer enn 30% av skjermen.
CSH'en må være brukervennlig, en ny bruker skal kunne finne informasjon på under 2/3 minutter.
CSH'en skal være optimalisert for pc og nettbrett
Systemets komponenter skal optimaliseres for gjenbrukbarhet hos NoIS
All systemdokumentasjon skal være forståelig, derav godt språk og gode formuleringer.
Systemet skal kunne utvikles ved hjelp av smidige utviklingsmetoder.

4.3.3 Ønskede funksjonelle krav:

CSH'en skal tilby brukeren å gjøre individuelle tilpasninger.
CSH'en skal inneholde en kursmodul der brukeren kan få en enkel opplæring av systemet.
CSH'en skal respondere på hjelpe-knapper i systemet.
CMS'en skal tilby funksjon for å eksportere CSH'ens innhold i sin helhet.
Søkefunksjonen skal bruke en AI til å analysere søkehistorikken for å forbedre søkeresultatene.
CSH'en skal som en del av support tilby en FAQ bot som benytter seg av søkefunksjonen sin AI.

4.4 Akseptansekriterier

Systemet anses som akseptert når de funksjonelle og ikke-funksjonelle kravene i kravspesifikasjonen er ferdig implementert og testet, og at testene selvsagt ikke avslører feil i systemet.

4.5 Teknologier og verktøy

Alle verktøy og programmeringsspråk er valgt basert på hva vi ønsker å benytte, med innspill fra oppdragsgiver basert på hva de bruker i en daglig praksis. Dette er gjort både for å skape et optimalt samarbeid, men også for å gi oss et innblikk i hva en bedrift bruker av teknologier og verktøy.

4.5.1 Programmeringsspråk og rammeverk:

VUE.js, HTML og SCSS:

Vue er et JavaScript bibliotek utviklet for å være lett å lære. Vue blir brukt på det visuelle laget av en applikasjon og har direkte kontakt med HTML. I tillegg til å brukes som en erstatter for JavaScript kan VUE brukes til å lage hele single-page applikasjoner. Vi vil bruke Vue sammen med SCSS og HTML for å lage vår applikasjon.

Vue test utils og Jest:

Jest er et JavaScript test rammeverk som har fokus på å være enkelt samtidig som det fungerer sammen med Vue. Jest har også en god dokumentasjon som gjør det enklere for oss å sette opp tester på en god måte. Vue test utils er det offisielle test biblioteket til Vue og er ofte brukt sammen med Jest.

4.5.2 Verktøy:

Valg av verktøy er gjort basert på at alle skal ha tilgang og kunne jobbe samtidig. Det er også valgt verktøy for å støtte en smidig utviklingsmodell.

VSCode:

VsCode(Visual studio code) er vår valgte editor, da denne støtter de aller fleste programmeringsspråk. Vi valgte denne editoren da den har støtte for Vue.js.

GitHub:

Vi bruker GitHub som versjonskontroll. Github gjør det mulig for oss å jobbe på kodebasen samtidig. Den gir god oversikt over hvem som har gjort hva og hva som er gjort. Vi ønsker å bruke branches for å kunne ha en enda bedre kontroll under utviklingen av vår applikasjon. Her knytter vi inn tankegangen fra CI, det at master branch bare blir benyttet til ferdig sjekket ut kode. All implementering/utvikling skjer på egne branches.

Google drive:

Vi bruker Google Drive til filer som ikke ligger på GitHub. Disse filene er regneark, rapporter, dagbok og diverse notater. Ved å bruke Google Drive får alle på gruppen mulighet til å skrive på dokumenter til en hver tid og samtidig.

Trello:

Trello er et kanban-board som blir brukt til å organisere prosjekt i et tavle-format. Dette verktøyet skal hjelpe med strukturering av oppgaver da man kan se hvem som jobber på hva og hva som jobbes med/er ferdig/mangler å jobbes på.

Microsoft Teams:

Teams brukes for kommunikasjon mellom oss i gruppen, og mellom gruppen og oppdragsgiver. Daglige møter og sprint reviews skal gjennomføres over Teams.

Adobe XD:

Adobe XD er et designverktøy for brukeropplevelse for web- og mobilapper. Verktøyet skal brukes for å designe brukergrensesnittet, samt lage klikkbare-prototyper av CSH'en. Adobe XD er ikke et utviklerverktøy som lager kode, men brukes til å lage en skisse av en ide for et bestemt produkt.

Microsoft Azure:

Vi bruker Azure som en skyplattform for infrastruktur og administrerte tjenester. Det er en cloud computing-plattform skapt av Microsoft, som omfatter virtuelle servere med nettverk, fillagring og databaser. Azure brukes fordi det er en skyplattform som er basert på åpen kildekode, som støtter et bredt spekter av operativsystemer, rammeverk og programmeringsspråk. Dette er også en plattform som ønskes implementert hos NoIS.

5. Løsninger / Alternativer

5.1 Vår løsning

Sammen med NoIS har vi vurdert ulike løsninger for hvordan en CSH skal utformes og implementeres slik at den kan tilby brukeren best mulig hjelp. Vi har kommet frem til en frontend komponent (CSH) som gir brukeren informasjon om den siden hen befinner seg på. CSH'en vil endre innholdet når bruker bytter side, og på den måten tilby en mer dynamisk hjelp. For at det skal være enkelt å legge inn innhold i CSH'en vil det også lages et publiseringssystem (CMS - Content Management System). Denne er med på å støtte tanken om at CSH'en enkelt skal kunne konfigureres til andre systemer. Siden det per dags dato ikke finnes noe ideelt system å teste og integrere CSH'en på, vil det også lages en dummy-applikasjon.

Det er sett på ulike løsninger for kommunikasjon mellom komponentene. Det store spørsmålet er om komponentene skal struktureres i en monolittisk eller microservice arkitektur. Med andre ord, om CSH'en skal være en integrert del av hovedapplikasjonen (som i våres tilfelle er en dummy), eller om CSH'en og hovedapplikasjonen er frittstående komponenter som kommuniserer gjennom API'er. Tabellen nedenfor beskriver de viktigste fordelene og ulempene ved de ulike arkitekturene.

Alternativ	Løsning	Fordeler	Ulemper
1	Monolittisk arkitektur	- Enklere å utvikle - Debugging og testing er enklere. - Enklere å gi ut, da alt kjøres samtidig.	- Krever at alt har samme teknologi-stack. - Vanskeligere å gjøre endringer, siden det kan påvirke hele systemet. - Store applikasjoner kan ta lengre tid å starte opp.
2	Microservice arkitektur	- Bedre skalerbarhet og fleksibilitet. - Feil i en komponent trenger ikke påvirke resten av systemet.	- Komponentene må kjøres individuelt. - Kommunikasjonen mellom komponentene blir mer komplisert. - Krever vesentlig mer kompetanse

5.2 Komponenter

5.2.1 Context-Sensitive Help (CSH)

Tanken er å gjøre komponenten bevisst på hvilken side i hoved applikasjonen bruker står på, for så å vise passende hjelp til bruker. Innholdet skal kunne være i form av både tekst, bilder og video. Innhold skal kunne deles inn i ulike kategorier som tips og ordforklaringer.
CSH'en skal også inneholde en søkefunksjon som gjør det lettere å finne spesifikk informasjon raskere. Vi har sett på ulike alternativer for hva søkefunksjonen skal søke gjennom, herav innholdet i CSH'en eller en brukermanual for systemet.

Alternativ	Løsning	Fordeler	Ulemper
1	Søkefunksjon søker gjennom innholdet til CSH'en	- Raskere.	- Kun avgrenset innhold.
2	Søkefunksjon søker gjennom hele brukermanual	- Alt innhold er tilgjengelig for søkefunksjonen.	- Tregere å søke gjennom hele dokumentasjonen.

Dersom CSH'en ikke klarer å tilby brukeren den hjelpen hen trenger vil systemet også tilby en snarvei til support. Her vil kontaktinfo være tilgjengelig og vi vil se på muligheten for å implementere en chatbot hvis vi får tid.

5.2.2 Content Management System (CMS)

For å enkelt konfigurere og oppdatere CSH'en vil det lages et publiseringssystem, der kun autoriserte brukere kan logge seg inn. Informasjonen som blir lagt inn blir så sendt til en database som CSH'en kan lese fra. Vi har sett på ulike løsninger for hvordan dette kan gjøres på en effektiv og mest mulig "fail-safe" måte. Alternativene består av å enten importere innhold fra PDF eller ved å fylle ut et skjema.

Alternativ	Løsning	Fordeler	Ulemper
1	Importere data fra PDF. Dette kan for eksempel gjøres ved hjelp av et verktøy som parser dokumentet.	- Raskt - Kan hente inn hele sider av gangen	- Dersom brukermanualen skal benyttes må den tilpasses først. Hvis ikke vil innholdet bli på et fagspråk. - Vil kreve kjennskap til verktøyet. - Vanskeligere å sortere ut i CSH. - Bruker third-party eks. Pdf2json. - Alternativ metode bør være tilgjengelig.
2	Bruk av skjema for å lage innhold	- Kan skreddersy innhold til CSH. - Enklere språk (hvis brukermanual benyttes i alt1) - Større muligheter i forhold til multimedia	- Krever manuelt arbeid for å skreddersy innhold.

5.2.3 Dummy Applikasjon

Dummy applikasjonen skal fungere som en svært enkel nettside hvor vi kan teste vår CSH, samt visualisere de funksjonaliteter som vil være aktuelle i OsloREG sitt system. En stor fordel ved å bruke en dummy applikasjon er at den ikke inneholder noen sensitive opplysninger om OsloReg. Resultatet kan dermed også brukes som en demo for NoIS under innsalg, og gruppemedlemme kan bruke den ved jobbsøking og lignende. Applikasjonen vil bestå av 3 sider med ulikt innhold. Alle sidene vil inneholde et header-element, et nøkkelord som kan fortelle oss at CSH'en viser riktig innhold. Forsiden vil bestå av 3 elementer med klikk-funksjonalitet. Ett element vil føre til side 2, ett annet vil føre til side 3 og det siste vil starte en fil-velger. På den andre siden vil man se et skjema for utfylling, samt en knapp som tømmer feltene i skjemaet uten å lagre innholdet. På side 3 vil det være en tabell med hard-kodet data. Dataen skal kunne sorteres basert på hvilken av headerne som blir trykket.

6. Analyse av virkninger

6.1 System arkitektur

En microservice arkitektur vil gjøre systemet svært distribuert, noe som blant annet gjør feilsøking og testing vanskeligere. Det er komplisert å sette opp backend, kommunikasjonen mellom komponentene, og man må håndtere uavhengige systemsvikter. Gruppemedlemmene har ingen erfaringer på området og det vil derfor være en stor risiko forbundet med å implementere microservices. Hovedutfordringen med en monolittisk arkitektur er at den vil begrense systemets funksjon som en hyllevare. Denne tradisjonelle arkitekturen krever at hovedapplikasjonen bygges i samme teknologi-stack som CSH'en, noe som med tiden vil gjøre den mindre og mindre kompatibel. Samtidig vil det være muligheter for å flytte komponentene over i en microservice arkitektur på et senere tidspunkt dersom det skulle bli behov for det. I så fall vil det være ekstra viktig at man tar høyde for dette under utviklingen.

6.2 Søkefunksjonen

Velger man å søke gjennom innholdet i CSH'en vil man søke gjennom innhold som man allerede har tilgjengelig, siden CSH'en skal vise informasjonen som er aktuell for den siden man befinner seg på. Det kan likevel være scenarioer der man er ute etter spesifikk informasjon om en del av systemet uten å kanskje vite hvor man finner denne informasjonen. I et slikt tilfelle vil søk på CSH'ens innhold være til stor hjelp. Alternativet, som er å søke gjennom en brukermanual, vil kunne tilby brukeren en mer utvidet og potensielt mer detaljert hjelp. Det forutsetter at brukermanualen er fullstendig og skrevet på et språk brukere forstår, uten tekniske fagbegreper ol. Her vil det være stor variasjon fra system til system.

6.3 CMS - PDF vs skjema

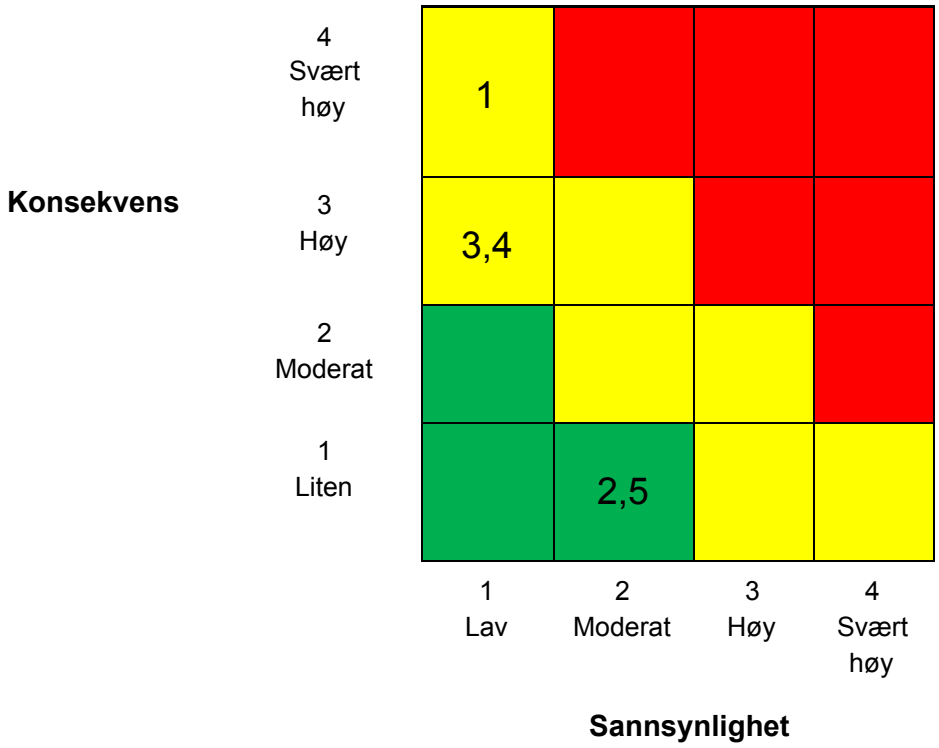
Dersom man velger å bruke et parser program for å generere innhold, vil man måtte følge strenge format regler på dokumentet som skal parses. Dette vil innebære betydelige ressurser til opplæring av de som skal produsere innhold til CSH'en. Det bør forøvrig også være mulighet for å gjøre endringer på resultatet av parseren, dersom den ikke skulle klare å parse innholdet riktig. Samtidig vil en PDF-parser være et raskere alternativ hvis format-regler blir fulgt og parseren har lav feilrate. Velger man i stedet å bruke skjema, vil labels indikere hvilken del av CSH'en brukeren fyller ut. På den måten vil det være enklere for hvem som helst å generere innhold, og føle hen har mer kontroll over hele prosessen.

7. Risikovurdering

Vi har gjort en risikovurdering av systemet i sin helhet som en kombinasjon av sannsynlighet for og konsekvensen av en uønsket hendelse. Denne systematiske gjennomgangen er essensiell for å vite hva som kan gå galt, finne ut hvordan vi forhindrer det, og hva vi må gjøre dersom noe likevel skulle inntreffe.

Nr.	Faktor	Sannsynlighet for risiko	Konsekvens av risiko	Tiltak
1.	Innhold er ikke beskyttet bak innlogging.	Liten	Kritisk	God nok autentisering, bruke ressurser på datasikkerhet.
2.	Ønsket informasjon ikke fremvist innen 5 sekunder.	Moderat	Liten	Lag god nok kildekode/algoritmer.
3.	Innhold bruker mer enn 30% av skjermen.	Liten	Alvorlig	Endre tilpasning kode ifht. skjermen.
4.	Systemet skal være plattformuavhengig.	Liten	Alvorlig	Produsere ny kode.
5.	Systemdokumentasjonen skal være enkel å forstå.	Moderat	Liten	Lage ny og forbedret dokumentasjon.

Risikoelementene vurderes langs to akser, - sannsynlighet og konsekvens. Systemet er på dette stadiet under utvikling og det er derfor blitt utført en vurdering av hele systemet, og ikke ulike komponenter i detalj. I våre analyser benytter vi en skala fra 1-4:



8. Fremdriftsplan

Fremdriftsplanen er utformet som et Gantt diagram. Diagrammet gir en oversikt over hvilke deler av prosjektet som skal jobbes med når og hvordan man ligger an underveis i utviklingen. Det gjør det også enklere å planlegge ulike aktiviteter i forhold til hverandre. F.eks har utviklingssprintene mye å si for hvordan man bør legge opp møter med oppdragsgiver.