Høgskulen på Vestlandet

BACHELOROPPGAVE

Timing informasjon

Timing Display

Niklas Vatshelle Raa, Thomas Thanh Cong Vu og Morten Tordal

Dataingeniør og informasjonsteknologi
Fakultet for ingeniør- og naturvitskap
Richard Kjepso
04.06.2021

Høgskulen på Vestlandet

Fakultet for ingeniør- og naturvitskap

Institutt for datateknologi, elektroteknologi og realfag

TITTELSIDE FOR HOVEDPROSJEKT

Rapportens tittel: Timing Informasjon	*Dato:* *04.06.2021*
Forfatter(e): Niklas Vatshelle Raa, Morten Tordal og Thomas Thanh Cong Vu	*Antall sider u/vedlegg: 30* *Antall sider vedlegg: 30*
Studieretning: Dataingeniør og Informasjonsteknologi	*Antall disketter/CD-er:* 0
Kontaktperson ved studieretning: Richard Kjepso	*Gradering:* Ingen
Merknader: Ingen	

Oppdragsgiver: Vizrt	*Oppdragsgivers referanse:*
Oppdragsgivers kontaktperson: Torbjørn Bøen	*Telefon:* 415 28 781

Sammendrag:

Denne rapporten handler om utviklingen av webapplikasjonen Timing informasjon. Webapplikasjonen er en videreutvikling av en tidligere applikasjon Vizrt har utviklet. Den nye løsningen er web-basert og har noen andre kriterier. Rapporten tar for seg utviklingsprosessen for webapplikasjonen, blant annet hvilke valg som ble tatt, problemer som oppsto underveis og en evaluering av resultatet.

This report is about the development of the web application Timing Display. The web application is a further development of a previous application developed by Vizrt. The new solution is web-based and has some other criteria. The report covers the development process for the web application, including the decisions made, problems that occurred in the process and an evaluation of the result.

Stikkord:

C#/.NET	Blazor WebAssembly	REST API

Høgskulen på Vestlandet, Fakultet for ingeniør- og natuvitskap
Postadresse: Postboks 7030, 5020 BERGEN Besøksadresse: Inndalsveien 28, Bergen
Tlf. 55 58 75 00 Fax 55 58 77 90 E-post: post@hvl.no Hjemmeside: http://www.hvl.no

FORORD

Denne rapporten tar for seg utviklingen av web-applikasjonen Timing Informasjon, heretter kalt ved originalt navn; Timing Display, våren 2021. Applikasjonen har tatt inspirasjon fra forrige versjon av Timing Display utviklet av Vizrt. Prosjektet er gjennomført av Niklas Vatshelle Raa, Thomas Thanh Cong Vu og Morten Tordal.

Gruppen ønsker å takke ekstern veileder Torbjørn Bøen for en utfordrende og spennende oppgave. Veileder har vært svært hjelpsom gjennom hele prosjektet, og har gitt gode tilbakemeldinger og rettledning på de punkter der gruppen har hatt problemer i utviklingen. I tillegg har veileder vært hjelpsom med å organisere testing av applikasjonen opp mot andre ansatte i Vizrt.

I tillegg retter vi også en takk til intern veileder ved Høgskulen på Vestlandet, Richard Kjepso for god oppfølging og hjelp underveis i prosjektet.

INNHOLDSFORTEGNELSE

1 INNLEDNING

1.1 Motivasjon og mål

Vizrt har utviklet en applikasjon som skal gi operatøren og nyhetsankeret timing informasjon under nyhetssendinger. Applikasjonen kalles Timing Display og er en Windows-basert WPF applikasjon. Windows Presentation Foundation (WPF) er et utviklingsrammeverk og et delsystem av .NET rammeverket, og brukes til å bygge Windows-klientprogrammer som kjører på Windows operativsystemet (Chaned, 2019). .NET rammeverket er en teknologi som støtter bygging og kjøring av Windows-apper og webtjenester (Microsoft, 2020). Applikasjonen gir brukeren informasjon og nedtellinger om aktuelle og kommende innslag i sendingen. Hovedmålet er å utvikle en konfigurerbar web-basert løsning som kan kjøre på alle plattformer inkludert mobil, nettbrett, datamaskin etc. Applikasjonen skal være enkel å bruke og forstå for journalister i felt.

Delmål for prosjektet:

- Iterasjon 1: Backend – frontend kommunikasjon.
- Iterasjon 2: Serialisere XML til tid-attributter.
- Iterasjon 3: Konvertere tid-attributter fra frames til sekunder.
- Iterasjon 4: Utvikle konfigurerbar frontend.

1.2 Kontekst

Viz Mosart er et verdensledende studio-automasjonssystem for produksjon av nyhetssendinger for kunder over hele verden. Under en slik nyhetssending er operatøren og nyhetsankeret helt avhengig av timing informasjon. Dette får de i dag ved hjelp av en applikasjon som kalles Timing Display. Problemet med dagens løsning er at applikasjonen kun kan kjøres på én Windows enhet, som egner seg dårlig for brukere som ikke befinner seg i studio. I tillegg kan den kun benyttes av en maskin om gangen. Hensikten er å gi kundene mulighet til å bruke en web-basert løsning med mulighet til å kjøre den på flere enheter samtidig, dette inkluderer mobil, nettbrett, datamaskin etc.

1.3 Avgrensninger

Prosjektet bygger på den gamle versjonen av Timing Display som er laget i Visual Basic .NET. Gruppen skal erstatte den gamle applikasjonen med en ny web-basert versjon. Det legges vekt på at brukergrensesnittet er konfigurerbart etter brukers ønske, slik at applikasjonen kan benyttes av personer med ulike roller. Hvordan applikasjonen skal distribueres og brukes blir opp

til Vizrt å bestemme når den er ferdig utviklet. Gruppen har ingen ansvar angående dette, og har kun i oppgave å utvikle applikasjonen.

Den pågående pandemien kan skape problemer for gjennomføringen av brukertestingen, grunnet at situasjonen endrer seg uke for uke, og det er vanskelig å legge planer om fysiske møter langt frem i tid. Det vil likevel være mulighet for å overlevere kode til eventuell testing, men dette kan føre til misforståelser og uforutsette problemer.

1.4 Ressurser

En av de viktigste ressursene til denne bacheloroppgaven er dagens Timing Display applikasjon. Denne applikasjonen tilfredsstiller de fleste kravene fra brukeren og blir en viktig ressurs ved utviklingen av prosjektet. En utfordring er at denne applikasjonen er skrevet i Visual Basic som gruppen ikke har erfaring med. Oppdragsgiver Torbjørn Bøen fra Vizrt er også en viktig ressurs for arbeidet med bacheloroppgaven. Gruppen ser for seg en jevnlig dialog med oppdragsgiver for å tilfredsstille kravene til kunden/brukerne. Oppdragsgiver har også erfaring med dagens Timing Display applikasjon og kan være en nyttig ressurs ved eventuelle spørsmål.

1.5 Oppbygging av rapporten

Kapittel 1	Presentasjon av oppgavens motivasjon og mål. Samt kontekst og avgrensninger til prosjektet. Hvilke ressurser som har vært tilgjengelig under prosjektet og rapportens oppbygging.
Kapittel 2	Presentasjon av bakgrunnen til problemstillingen.
Kapittel 3	Prosjektets design, løsningsalternativer, valgt løsning og prosjektmetodikk.
Kapittel 4	Stegene for å oppnå prosjektets ulike resultater.
Kapittel 5	Hvilke evalueringsmetoder har blitt brukt, og hva er resultatet av disse metodene.
Kapittel 6	Resultatene fra evaluering fra ekstern veileder og brukertester.
Kapittel 7	Konsekvensene av gruppens tilnærming til det oppnådde målet, valgenes påvirkning på resultatet og prosjektets forbedringspotensialer.
Kapittel 8	Oppsummering av prosjektets mål og delmål og forslag til videre arbeid.
Kapittel 9	Referanser
Kapittel 10	Risikoliste, GANTT diagram og brukermanual

2 PROSJEKTBESKRIVELSE

2.1 Praktisk bakgrunn

Vizrt har vært involvert i tidligere prosjekter med Høgskulen på Vestlandet, men gruppen og oppdragsgiver har ikke samarbeidet noe tidligere. Kommunikasjon og fremgang med prosjektet har ikke blitt påvirket av dette, og gruppen har fått rede på initielle planer og krav på hvordan applikasjonen skal fungere og se ut når den er ferdig.

2.1.1 Prosjekteier

Vizrt er verdensledende leverandør av visuelle historiefortellingsverktøy for medieinnholdsskapere innen kringkastings-, sports-, digital- og e-sportsindustrien. Vizrt tilbyr markedet definerende programvarebaserte løsninger for sanntids 3D-grafikk, videoavspilling, studioautomatisering, sportsanalyse, medieformuesforvaltning og journalist historiske verktøy. Vizrt sitt løfte er å mestre kompleksitet og maksimere kreativiteten. Mer enn tre milliarder mennesker ser historier fortalt av Vizrt-kunder hver dag, inkludert fra medieselskaper som CNN, CBS, NBC, Fox, BBC, BSkyB, Sky Sports, Al Jazeera, NDR, ZDF, Network 18, tencent og mange flere.

2.1.2 Tidligere arbeid

Vizrt har i dag en Windows-basert WPF applikasjon, som nevnt i kapittel 1.1 brukes WPF til å bygge Windows-klientprogrammer som kjører på Windows operativsystemet, med navn Timing Display. Applikasjonen er i bruk av Vizrt sine kunder over hele verden. Det er derfor ingen tvil om behovet for en slik applikasjon, men dagens løsning har noen begrensninger. Den kan kun kjøres på en Windows enhet, så journalister har ikke hatt muligheten til å bruke applikasjonen for å få tidsinformasjon når de er ute i felt.

2.1.3 Initielle krav

Vizrt sine initielle krav er at det skal utvikles en webapplikasjon som skal ha like funksjoner som den tidligere applikasjonen, men at den skal kunne bli kjørt på alle plattformer. Løsningen skal være konfigurerbar og brukervennlig. Webapplikasjonen skal kommunisere med et REST API, som er en tilstandsløs klient-server kommunikasjon hvor man sender forespørsler gjennom HTTP metoder (RedHat, 2021) basert på .NET Core Web API. Det skal utvikles algoritmer for utregning av nedtellingene.

2.1.4 Initiell løsnings-idé

Målet er å lage en konfigurerbar real-time HTML5-basert løsning, fra bunnen av, som kan kjøre på alle plattformer inkludert mobil, nettbrett, datamaskin etc. Gruppen skal utvikle et enkelt brukergrensesnitt slik at brukerne kan lett lære og forstå hvordan man skal bruke applikasjonen. Applikasjonen skal kommunisere med Viz Mosart og bestå av et konfigurerbart rutenett med ulike felt som man kan tilpasse for eget bruk. På disse feltene skal man kunne endre på:

- Størrelse
- Posisjon
- Tekst
- Tid

Figur 2-1: Skjermbilde av dagens Timing Display applikasjon

2.2 Litteratur om problemstillingen

Litteratur som er brukt som kilder til denne oppgaven er hentet fra nettsteder og elektroniske bibliotek. Informasjon og beskrivelse av rammeverk og ressurser som er brukt er hentet direkte fra leverandør eller utgiver.

Figur 2-2: Codecademy logo

På Codecademy kan man finne mange ulike kurs for å lære ulike kodespråk. Siden gruppen ikke har erfaring med C# og Blazor tidligere, så har det hjulpet å gå gjennom kurs for C# og ASP.NET hvor en går gjennom C# og bygging med ASP.NET Core og Blazor.

(Codecademy, 2021)

Figur 2-3: Blazortrain logo

Blazortrain har videoer om alt en skulle trenge for å kunne lære og forstå seg på Blazor fra start til slutt.

(Blazortrain, 2020)

3 DESIGN AV PROSJEKTET

3.1 Forslag til løsning

Et av kravene fra oppdragsgiver er at applikasjonen skal kommunisere med et REST API. REST API-et defineres som applikasjonens backend til prosjektet og skal kobles opp mot en annen applikasjon Vizrt har utviklet, som heter Mosart. Grunnet dette kravet er applikasjonens backend bundet til å utvikles i .NET rammeverket.

Et annet krav fra oppdragsgiver er at applikasjonen skal kunne kjøres på flere plattformer. Applikasjonen må derfor kunne kjøres i nettleseren på ulike enheter og størrelser, men gruppen står fritt til å velge rammeverk for å tilfredsstille dette kravet. Det er to alternativer som settes i fokus, og de skal utforskes og evalueres med tanke på fordeler og ulemper.

3.1.1 Alternativ løsning 1 – Frontend i JavaScript

Den første løsningen er å utvikle frontend i JavaScript. JavaScript er sammen med HTML og CSS de tre kjerneteknologiene for utvikling av websider. JavaScript har flere frontend rammeverk som React, Vue, Angular etc. Det er med andre ord svært god dokumentasjon på frontend utvikling i JavaScript.

3.1.2 Alternativ løsning 2 – Frontend i .NET

En annen løsning går ut på å utvikle frontend i .NET. .NET er en samling av teknologier der Blazor er spesifisert for utvikling av webapplikasjoner. Blazor gjør det mulig å utvikle webapplikasjoner med HTML, CSS og C#. (Microsoft, 2021)

3.1.3 Diskusjon av alternativene

Alternativ løsning 1 går ut på å velge det rammeverket gruppen har mest erfaring med. Gruppen har gjennom studiet lært seg JavaScript og har mest erfaring med det. Utfordringen her er at gruppen har for det meste utviklet enkle UI tidligere. Dette prosjektet blir derfor mye mer kompleks enn det gruppen har erfaring med.

Alternativ løsning 2 går ut på å velge det rammeverket oppdragsgiver kan og bruker. Fordelen med å velge .NET som rammeverk er at gruppen kan få god hjelp og både frontend og backend utvikles i samme rammeverk. I tillegg må backend for prosjektet utvikles i .NET rammeverket. Det betyr at hele prosjektet kan samles i Visual Studio slik at gruppen har enkel oversikt over de ulike delene. Utfordringen med denne løsningen er at gruppen har ingen erfaring med .NET og

programmeringsspråket C#. Det vil si at gruppen må bruke mye tid på å lære seg det grunnleggende før de kan begynne å utvikle applikasjonen.

3.2 Valgt løsning

Alternativ løsning 2 ble valgt ettersom gruppen konkluderte og ble enige om at dette var det mest gunstige løsningen. Fordelene ved denne løsningen ble ansett som de viktigste og ulempene er på ingen måte kritiske. Det å kunne få god hjelp dersom gruppen eventuelt skulle stå fast ved det tekniske vil være svært nyttig i utviklingsprosessen. Den valgte løsningen krevde at gruppen brukte litt tid i oppstarten på å lære seg .NET og programmeringsspråket C#, men gruppen så ikke for seg at dette skulle være problematisk.

3.3 Valg av verktøy

I dette prosjektet benyttes følgende verktøy:

- Visual Studio 2019 - IDE kode editor (Microsoft, 2021)
- .NET rammeverk - Utvikler plattform (Microsoft, 2021)
- C# - .NET programmeringsspråk (Microsoft, 2021)
- Blazor WebAssembly - Bygge klientbaserte webapplikasjoner med C# (Microsoft, 2021)
- HTML og CSS - To kjerneteknologier for utvikling av websider (W3C, 2016)
- GitLab - DevOps plattform (GitLab, 2021)

Noen av verktøyene er valgt ut fra oppdragsgivers krav, og andre basert på hensiktsmessighet og i forbindelse med valg løsning.

3.4 Prosjektmetodikk

3.4.1 Utviklingsmetodikk

Under prosjektutviklingen benytter gruppen seg av en Agile arbeidsmetode. “Agile arbeidsflyt er en iterativ metode for å levere et prosjekt. I denne arbeidsmetoden jobber flere lag på spesifikke oppgaver for en bestemt varighet som kalles ‘sprinter’.” (Petkar, 2020).

Det finnes flere typer for Agile arbeidsmetoder, og gruppen anser scrum som den mest passende for prosjektet. Denne typen legger vekt på kontinuerlig forbedringer for kundens tilfredshet. Det er avtalt ukentlige statusmøter med oppdragsgiver hvor det blir gått gjennom hva som har blitt gjort siden sist og det settes mål for neste statusmøte. Dette passer veldig bra med den Agile arbeidsmetoden, slik at gruppen har sprinter med mål som skal bli gjennomført hver uke. Dette

fører også til tett samarbeid med oppdragsgiver slik at produktet blir mest mulig lignende det de ser for seg.

3.4.2 Prosjektplan

Planen for prosjektet startet i dialoger sammen med oppdragsgiver. Sammen ble det gjennomgått en prosjektplan og gruppen fikk satt opp et foreløpig Gant-diagram. “Et Gantt-diagram er en teknikk som brukes for å visualisere prosjekters tidsplan med aktiviteter og milepæler.” (Wålberg, J. A., 2020) Prosjektplanen skal i hovedsak følge førsteutkastet av Gantt-diagrammet så godt det lar seg gjøre, men det er mulig å gjøre endringer dersom det blir behov for det.

3.4.3 Risikovurdering

For å vurdere risiko har gruppen satt opp en risikomatrise for alle mulige risikoer. I matrisen diskuteres hver risiko etter sannsynlighet og konsekvens. Alvorligheten blir regnet ut ved å multiplisere sannsynlighetsgraden og alvorsgraden. Til slutt tenkes det frem forskjellige løsninger på hvordan disse risikoene kan håndteres. Disse løsningene ligger i siste kolonne av risikomatrisen.

3.5 Evalueringsplan

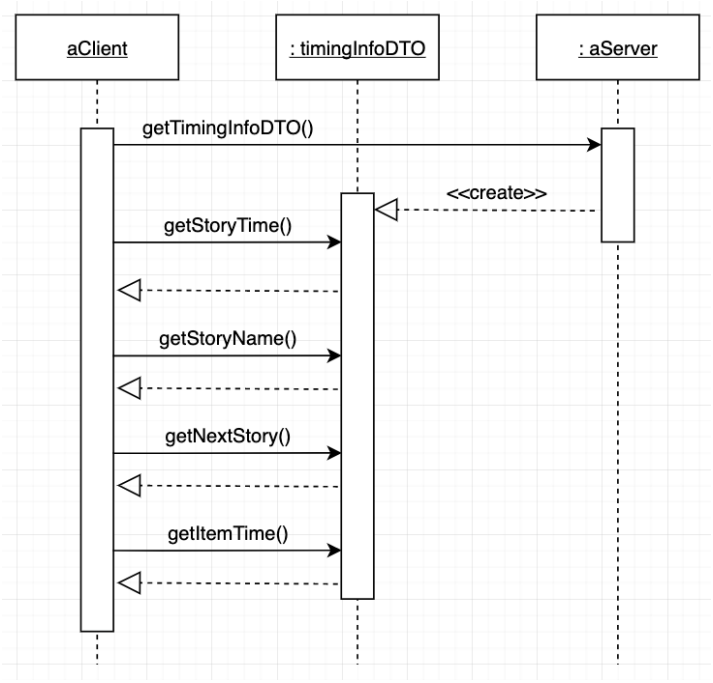
Under de ukentlige statusmøtene med oppdragsgiver kan oppdragsgiver evaluere produktet kontinuerlig underveis i produktutviklingen. Kontaktperson vet hvilke krav produktet skal ha og kan gi gruppen tilbakemeldinger på om dette er en god løsning for brukerne. I tillegg tilbyr oppdragsgiver brukertesting av applikasjonen. Oppdragsgiver kommer da til å kontakte en mengde journalister som kan teste applikasjonen i relevant jobbsammenheng. Dette vil foregå i sluttfasen når produktet skal finjusteres.

4 DETALJERT DESIGN

4.1 API

Et av de viktigste punktene for at applikasjonen skal fungere er et velfungerende API. API-et i prosjektet er REST-basert. Det vil si at det er en tilstandsløs klient-server kommunikasjon hvor man sender forespørsler gjennom HTTP metoder (RedHat, 2021). API-et i prosjektet er utviklet basert på .NET Core Web API. Det er kun utviklet én HTTP-metode i API-et, som er en HTTP GET metode. Metoden henter en "heartbeat" fra Mosart. "Heartbeaten" er en samling av alle navn, tider og nedtellinger som er relevant for applikasjonen.

HTTP metoden returnerer "heartbeaten" i XML format. Siden gruppen ønsker å jobbe med tidene før det vises frem til brukeren har gruppen har implementert en serialiserer for å gjøre om innholdet i "heartbeaten" til en DTO (Data Transfer Object). En DTO benyttes for å overføre data mellom prosesser (Microsoft, 2014). På denne måten kan gruppen gjøre kall på DTO-klassen lokalt i stedet for å kalle på API-et hver gang. Det er da mulig å hente «Story Time» direkte fra DTO-klassen og skrive det ut til klienten. Hvordan dette gjennomføres demonstreres i figuren nedenfor.



Figur 4-1: Sekvensdiagram Klient-DTO-Server

4.2 Algoritmer for utregning av nedtellingene

Applikasjonen skal vise ulike tider med ulike formateringer. Noen skal være nedtellinger, hvor noen vises med timer og andre kun med minutter og sekunder, og andre skal være formatert i form av klokkeslett. Etter et møte med Vizrt ble vi enige om at klokkeslett skal vises med 12-timers klokke med AM og PM istedenfor 24-timers klokke. Når vi henter tidene fra API-et, er de ikke i det formatet vi ønsker å vise fram til brukeren. De blir hentet i form av frames som er vanskelig for brukeren å tolke, derfor må de gjøres om til timer, minutter og sekunder for å gjøre tidene mer lesbar for brukeren.

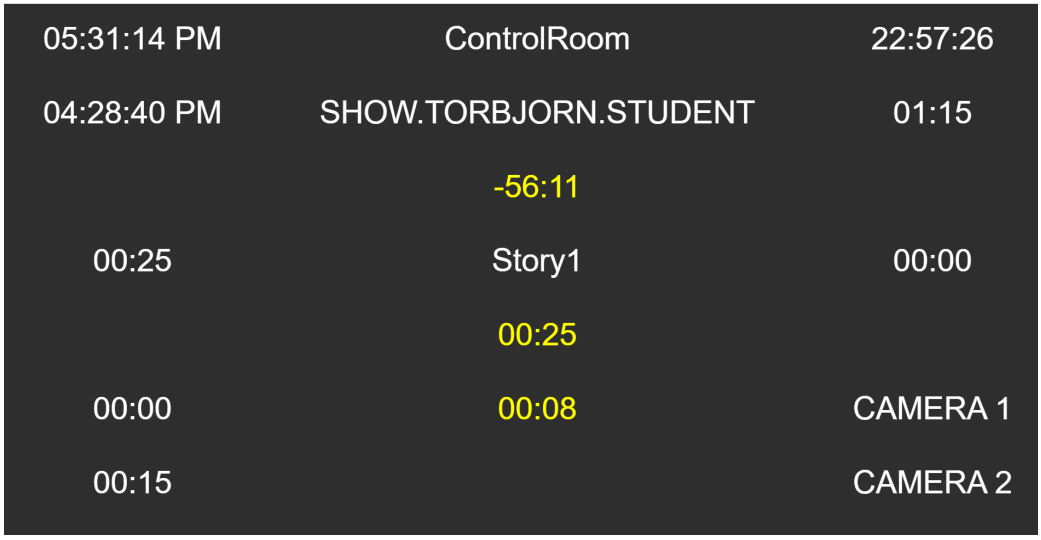
Det er også noen viktige tider som gruppen ikke får direkte fra API-et. Gruppen har utviklet algoritmer for utregningene av disse. I tillegg skal noen av nedtellingene vises med et minus eller pluss fortegn når nedtellingen har passert null. Gruppen har også utviklet algoritmer for å formattere disse tidene riktig.

4.3 Brukergrensesnitt (User Interface)

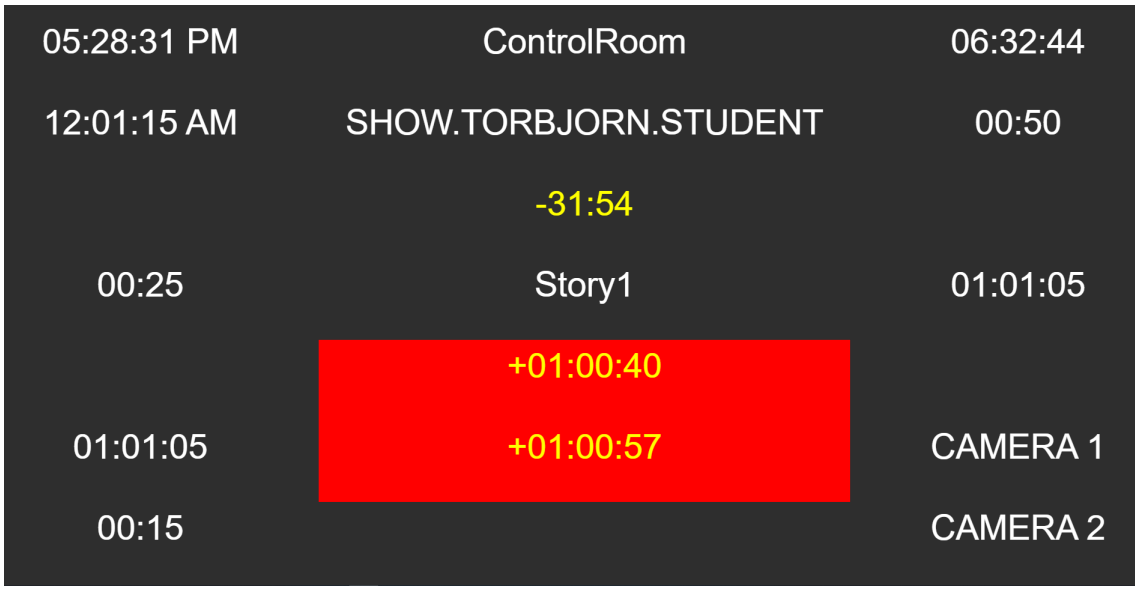
Det viktigste punktet i prosjektet er å utvikle et brukergrensesnitt som oppfattes brukervennlig og kan brukes på flere plattformer. Det er et viktig kriterium fra oppdragsgiver og noe gruppen har lagt mye vekt på.

4.3.1 Design

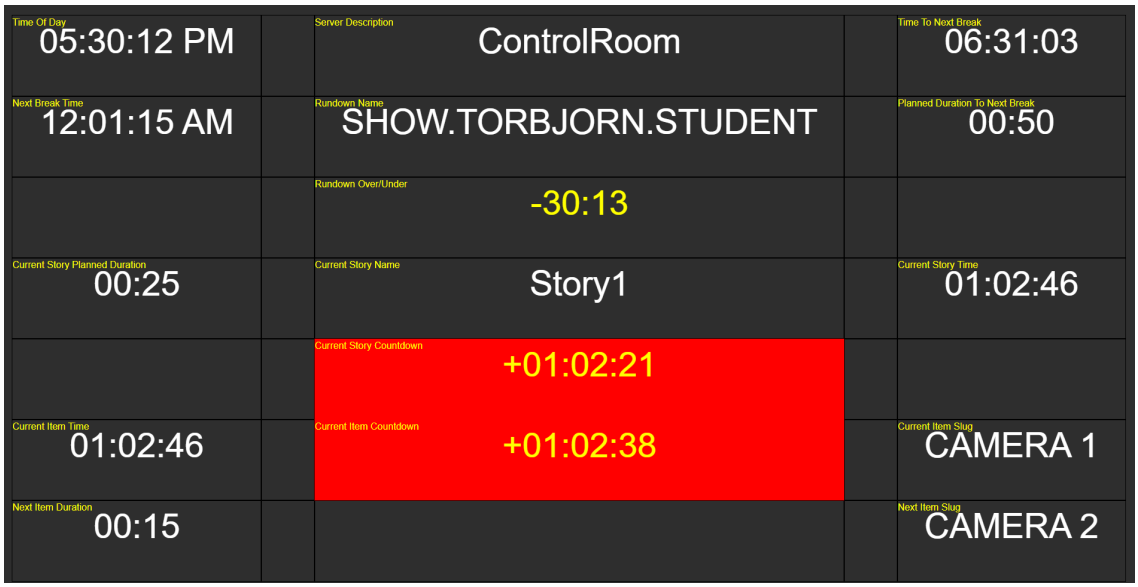
I arbeidet med brukergrensesnittet har design vært mye i fokus. Grunnet applikasjonen skal brukes som et alternativ sammen med dagens timing display falt beslutningen på å ha mye lignende design på prosjektet som dagens timing display. Det blir derfor mindre forvirring blant Vizrt sine kunder når de tester applikasjonen for første gang. Figurene nedenfor viser designet på brukergrensesnittet.



Figur 4-2: Skjermbilde av applikasjonen, datamaskinversjon

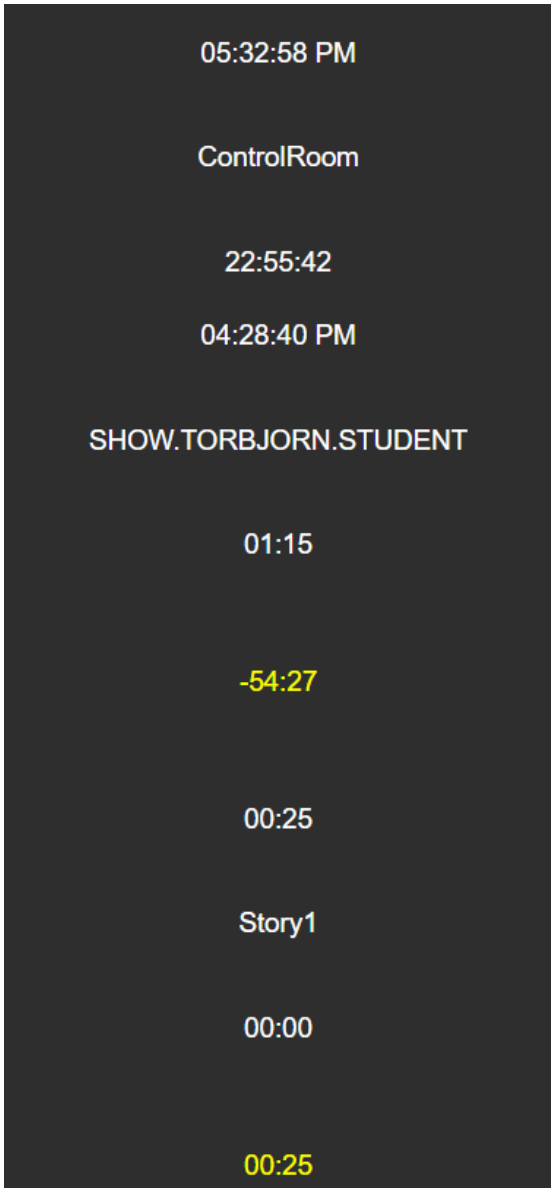


Figur 4-3: Skjermbilde av applikasjonen, datamaskinversjon med fargeendring



Figur 4-4: Skjermbilde av applikasjonen, datamaskinversjon med rammer og merkelapp

Et av kriteriene til applikasjonen er at den skal kunne brukes på flere plattformer. Gruppen har tatt hensyn til dette og utviklet et responsivt design som fungerer til alle skjermstørrelser. Figuren under viser hvordan applikasjonen ser ut med skjermstørrelsen til en mobiltelefon.



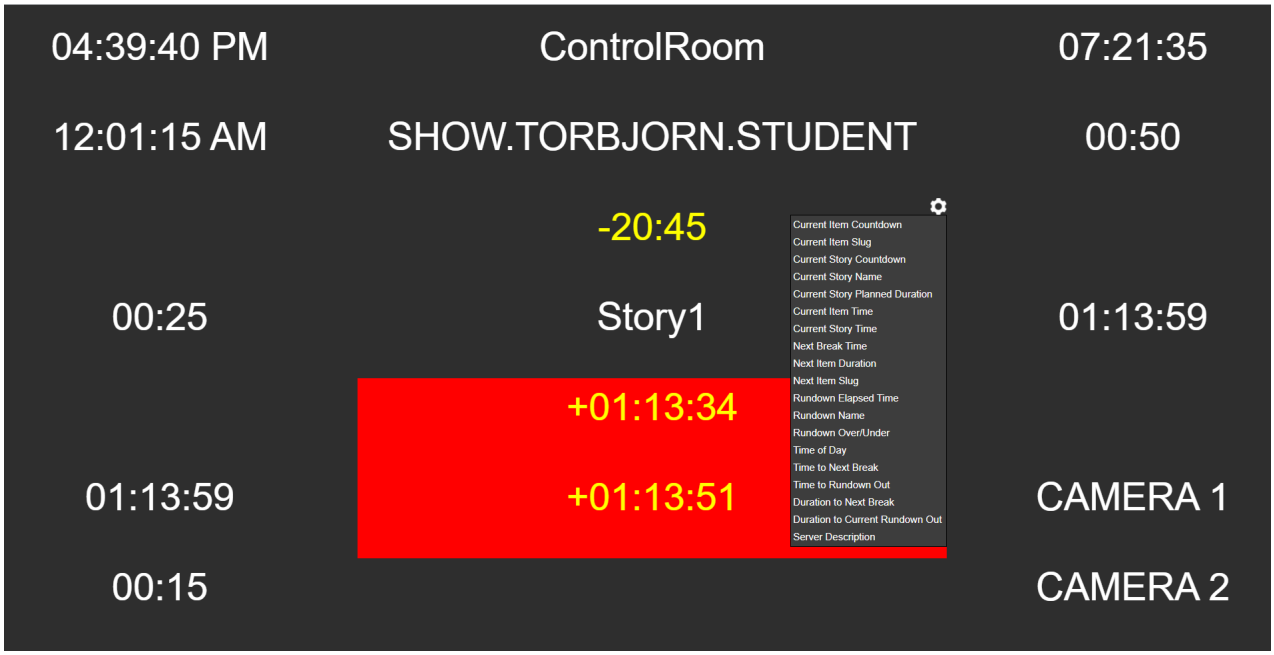
Figur 4-5: Skjermbilde av applikasjonen, mobil-versjon

4.3.2 Drag & drop

En viktig funksjon for applikasjonen er drag & drop. Hensikten for denne funksjonen er at brukeren skal ha mulighet til å kunne dra de ulike tidene i feltene de selv ønsker etter sitt behov. En skal kunne dra hvilken som helst tid til et tomt felt, og hvis feltet allerede inneholder en tid, så bytter disse to plass. Størrelsen på feltet endres etter størrelsen på tiden. Når bruker drar på en tid, skal det vises rammer på grid og indikator på hver tid.

4.3.3 Nedtrekksmeny

Når bruker holder musepekeren over en celle, vises et tannhjul øverst i høyre hjørne i cellen. Ved å trykke på tannhjulet vises en nedtrekksmeny. I denne menyen så skal brukeren kunne velge mellom å vise og skjule en tidsinformasjon. I tillegg skal bruker kunne velge å vise annen informasjon i dette feltet. Dette er en funksjon som vil være nyttig fordi en person som sitter i studio og styrer tv-sendingen har behov for oversikt over flere tider enn en reporter som er ute på oppdrag. En reporter i felt trenger kanskje bare et fåtall tider for å utføre jobben sin, mens personer i studio har behov for langt mer informasjon.



Figur 4-6: Skjermbilde av applikasjonen med nedtrekksmeny

5 EVALUERING

Prosjektets evalueringsplan handler i første omgang om å ha tett dialog med arbeidsgiver. Ved de ukentlige statusmøtene med oppdragsgiver kan oppdragsgiver evaluere produktet kontinuerlig underveis i produktutviklingen. Kontaktpersonen vet hvilke krav produktet skal ha, og har dermed gitt gruppen tilbakemeldinger på om utviklet løsning har vært god nok for fremtidige brukere. I tillegg tilbyr oppdragsgiver brukertesting av applikasjonen. Den opprinnelige evalueringsplanen av applikasjonen var at den skulle testes av journalister gjennom relevant jobbsammenheng, grunnet den pågående COVID-19 pandemien har dette ikke vært mulig. Derfor er det planlagt en alternativ evaluering av applikasjonen hvor ansatte i Vizrt som har jobbet som journalister tidligere skal teste bruken på en tilsvarende måte som de ville gjort dersom de faktisk var på jobb, enten som journalister eller i studio under en nyhetssending.

5.1 Evalueringsmetode

Evaluering av designet er gjort kontinuerlig gjennom utviklingen. Ekstern veileder har forklart for gruppen hvordan den tidligere versjonen av Timing Display har vært designet og hva som har vært foretrukket av brukerne. Det har da blitt bestemt av bakgrunnen skal være svart og teksten skal være hvit. Det er også et par unntak der noen tider som betraktes som mer viktig skal merkes gult. Dette gjelder for tidene rundownOverUnder, currentStoryCountdown og currentItemCountdown.

En av de viktigste metodene for utviklingen har vært prøve og feile metoden. En måte å oppnå et mål eller løse et problem ved å prøve flere forskjellige metoder og lære av feilene som blir gjort (Cambridge Dictionary, 2021). Dette har vært den mest sentrale metoden vi har brukt gjennom hele prosjektet. Ingen av gruppemedlemmene hadde særlig erfaring med C#/.NET før prosjektets start, så dette har vært en ofte brukt metode, både for læring av språk/API og testing av applikasjon.

I tillegg til prøve og feile metoden har kontinuerlig funksjonstesting med feilsøking vært en viktig faktor for utviklingen. Gruppen har ved implementering av nye metodekall funksjonstestet at disse fungerer som planlagt, og dersom de ikke har gjort det har det blitt gjort feilsøking for å finne eventuelle feil og mangler. Gruppen benyttet Swagger for å teste at API-et var riktig i starten av utviklingen. Swagger tester API-kall, for å sjekke om de fungerer som de skal.

5.2 Evalueringsresultat

Designet og funksjonene har blitt evaluert kontinuerlig gjennom hele prosjektet og gruppen har til enhver tid vært innforstått med utformingen av applikasjonens brukergrensesnitt. I motsetning til den originale versjonen som består av 5x8 kolonner, består den nye applikasjonen av 5x7 kolonner. Dette er besluttet for å gi en bedre brukeropplevelse, om dette medfører riktighet vil vise seg under brukertestingen. Ved å lage en ny applikasjon basert på det gamle designet vil med stor sikkerhet gi en brukeropplevelse som er nokså lik slik den var. I tillegg har tilbakemeldingene fra veileder gjort det mulig å gjøre større endringer i designet uten at dette har påvirket utviklingsprosessen i negativ grad i form av at ting ikke har fungert som ønsket etter at endringer har blitt foretatt.

Underveis i utviklingen har prøve og feile metoden, sammen med funksjonstesting for å se at de ulike metodene i applikasjonen fungerer som ønsket og feilsøking for å avdekke eventuelle feil vist seg å gi en del utfordringer.

Bruken av Blazor for å utvikle interaktive web brukergrensesnitt istedenfor JavaScript (Microsoft, 2021) har vært utfordrende for gruppen siden dette har vært et helt nytt rammeverk, i tillegg til at C# var et språk som var ubenyttet fra før hos alle gruppemedlemmene. Dette har ført til mer prøving og feiling enn det som ville vært nødvendig i et rammeverk og språk gruppen var kjent med og hadde erfaring med fra før. Dette har forbedret seg etter hvert som utviklingen har funnet sted og medlemmene av gruppen har fått mer erfaring med gjeldende rammeverk og språk.

Tidlig i utviklingen oppsto det problemer med heartbeaten. Problemet gjorde at applikasjonen ikke lastet riktig ved oppstart og krevde en oppdatering av siden for å fungere normalt. Dette løste gruppen ved feilsøking, for å finne ut at en service applikasjonen benytter, ikke ble opprettet ved oppstart av applikasjonen. Gruppen oppdaget også enda et problem med heartbeaten. Etter en stund stoppet heartbeaten, som førte til at applikasjonen ikke lengre fungerte som den skulle. Etter dialog med oppdragsgiver viste denne feilen seg som en bug i oppdragsgivers kode. Dette fikset oppdragsgiver opp i selv.

I utviklingen av frontend-delen av prosjektet brukte gruppen Blazor WebAssembly. Det ble utviklet et datagrid for å vise frem tidene og nedtellingene. Gruppen klarte i tillegg å implementere drag & drop, som ga et delvis oppfylt delmål 4, en konfigurerbar frontend. Videre i utviklingen oppsto det vanskeligheter får å løse andre funksjonaliteter, som visning av rammer og merkelapper. Dette ble løst ved å bruke JavaScript interop, som kaller på JavaScript fra .NET (Blazor University, 2021).

6 RESULTATER

Ekstern veileders kontinuerlige oppfølging én gang i uken har ført til at utviklingen hele tiden har fulgt de mål og ønsker Vizrt har hatt for det ferdige produktet. Evalueringen har bestått av en presentasjon av endringer og oppdateringer fra siste ukes arbeid. Deretter har gruppen fått hjelp til å løse diverse utfordringer og problemer som ikke har latt seg løse på egenhånd. Etter at problemer og spørsmål er utbedret, har veileder i samarbeid med gruppen kommet til enighet om hva som skal gjøres til neste ukes møte. Dette har gitt gruppen en god oversikt og forståelse av hva som må gjøres hver uke og det har hele veien vært enighet arbeidet gjennomføring og resultat.

Som et resultat av denne typen veiledning har det gitt en utvikling som har fulgt en ganske rett linje. Det har ikke vært noen betydelige sidesprang der gruppen har utviklet noe som ikke har vært ønsket av oppdragsgiver. Gruppen har nok spart mye tid på å ikke drive med unødvendig utvikling. I tillegg har gruppen hele tiden vært sikker i utviklingen og hva som skal gjøres. Det har vært lite spørsmål innad om det aktuelle arbeidet faktisk er noe som er ønsket eller om det går inn under unødvendig utvikling.

Brukertesting ble utført digitalt av en ansatt i Vizrt, som tidligere har jobbet i TV2 og har god erfaring med bruk av Mosart. Vedkommende har ikke benyttet Timing Display i stor grad tidligere, men er kjent med hensikten og bruken av applikasjonen. I starten av testingen ble nedtrekksmenyen misforstått. Bruker trodde at ved å trykke på den ønskede tiden, så ble den skjult på den aktuelle plasseringen til denne tiden. Bruker misforsto muligheten til å synliggjøre den aktuelle tiden et annet sted på skjermen. Etter hvert forsto bruker at han kunne benytte nedtrekksmenyen til å flytte informasjonen i cellene og tilpasse oppsettet etter eget ønske, dette som et alternativ til å dra og slippe celler. Under testingen brukte bruker lang tid på å forstå at indikatorene til hver celle ble synlig ved å holde inne musetasten. Derfor kan det tenkes at et valg om å ha disse synlig eller skjult etter brukers ønske, ikke bare ved flytting av tider er nødvendig. Dette for å forbedre brukeropplevelsen til en bruker som gjerne ikke har mye erfaring og eller kontroll på hva den ulike informasjonen i cellene betyr, samtidig som det gir en bedre oversikt over innholdet på skjermen.

7 DISKUSJON

En beslutning som ville vært annerledes ved en gjentagelse av prosjektet ville vært at alle gruppens medlemmer hadde tilgang på Windows-maskiner. Slik at alle gruppens medlemmer hadde mulighet til å jobbe med utviklingen på lik måte. Slik prosjektet ble gjennomført var det to av tre som hadde Windows og dermed kunne jobbe med både backend og frontend. Dette fordi applikasjonen gjør metodekall mot Mosart, som kun kan kjøres på Windows. Det tredje medlemmet hadde kun tilgang på Mac og dermed også kun mulighet til å arbeide med frontend.

Gruppens valg av rammeverk som oppdragsgiver både kan og selv bruker har vist seg å være svært nyttig under utviklingen. Sammen med de kontinuerlige møtene med oppdragsgiver har det stort sett alltid blitt gitt et konkret svar eller rettledning på spørsmål og vanskeligheter. I tillegg har "Agile" arbeidsmetode passet veldig godt under utviklingen og kundens tilfredshet, i dette tilfelle veileder, har blitt evaluert hver uke og det har kontinuerlig blitt satt nye utviklingsmål til påfølgende uke. Dette har ført til et tett samarbeid som for gruppen har blitt opplevd som svært positivt fordi den får respons på arbeidet som blir gjort med korte mellomrom. Noe som også har gjort mulighetene for eventuelle endringer mindre problematisk enn den hadde vært om det var større intervall mellom møtene. Gode tilbakemeldinger og støtte fra veileder har gjort at gruppen er svært godt fornøyd med valg av løsning 2 for prosjektet. Å lære seg de aktuelle rammeverk og språk har vært en grei oppgave og samtlige gruppemedlemmer har opparbeidet seg et godt grunnlag og god forståelse av disse. Sammen med tilbakemeldingene har dette gjort utviklingen smidig og rask. I motsetning til hva den kunne vært om alternativ 1 for løsning hadde blitt valgt, der veileder kanskje hadde hatt færre innspill og støtte å komme med.

Det har også blitt samarbeidet med andre bachelorgrupper som arbeider med samme rammeverk og programmeringsspråk når det har vært noe gruppen ikke har klart å løse på egenhånd. Dette har gitt god effekt både resultatmessig og læringsmessig, så her er gruppen fornøyd med resultatene av dette valget.

Strategien om å følge GANTT-diagrammet har fungert bra. Det har vært noen endringer i når de ulike delmålene har hatt tidsfrist, grunnet den manglende erfaringen gruppen hadde med utvikling i .NET og C#.

En begrensing som har tatt mye tid under utviklingen er Visual Studio 2019s manglende mulighet for live preview under koding, noe som ifølge Microsoft skal være en mulighet når .NET6 lanseres. Dette vil utgjøre en betydelig forbedring av tidsbruken under

utviklingen, ved å slippe å kjøre applikasjonen på nytt for hver eneste gang det er foretatt en endring i koden vil tidsbruken gå ned betydelig.

Dersom vi fikk muligheten til å gjennomføre prosjektet på nytt vil et naturlig svar være å kun bruke de verktøy som til slutt ledet til et ferdig produkt og ikke røre noe av det som fungerte.

8 KONKLUSJON OG VIDERE ARBEID

Hovedmålet er å utvikle en konfigurerbar web-basert applikasjon som gir brukeren informasjon og nedtellinger om aktuelle og kommende innslag i en tv-sending, i tillegg skal den kunne kjøre på alle plattformer, inkludert mobil, nettbrett, datamaskin etc. Applikasjonen skal være enkel å bruke og forstå for journalister i felt.

Delmål for prosjektet:

- Iterasjon 1: Backend - frontend kommunikasjon.
- Iterasjon 2: Serialisere XML til tid-attributter.
- Iterasjon 3: Konvertere tid-attributter fra frames til sekunder.
- Iterasjon 4: Utvikle konfigurerbar frontend.

Ved avslutning av prosjektet har gruppen oppnådd de tre første delmålene, og til dels nummer fire. Iterasjon 1, backend og frontend kommuniserer og tidene som hentes fra Mosart vises på skjermen når Timing Display applikasjonen kjører. Iterasjon 2, tidene applikasjonen henter fra Mosart gjøres om til attributter i en DTO-klasse. Iterasjon 3, konvertere tid-attributter fra frames til sekunder er implementert med noen hjelpeklasser. Det er én hjelpeklasse for å konvertere tider til riktig format og én hjelpeklasse for å gjøre utregninger. Iterasjon 4, bruker kan flytte tider til foretrukket plass på skjermen, i tillegg til å endre hvilken tid som skal vises i en celle.

Gruppens resultater vil videre kunne være til nytte for andre grupper med tilsvarende problemstillinger og oppgaver. Som nevnt i kapittel 7 har gruppen samarbeidet med en annen bachelorgruppe med lignende problemstilling når det har oppstått vanskeligheter og problemer vi ikke har funnet en løsning på alene. I tillegg kan vårt grunnlag være et godt grunnlag for eventuell videreutvikling, både for utviklere med god erfaring, men også mindre erfaring fra de ulike rammeverkene og språkene. Siden mye av tiden har gått med til å finne algoritmer for utregningene av de ulike tidene, vil det være mer tid til å fokusere på frontend og brukeropplevelsen i fremtidig utvikling.

Dersom arbeidet videreføres fra Vizrt sin side er det noen tider som bør implementeres og rettes for å få en 100% ferdig utviklet applikasjon. Grunnet begrensninger i ressursene gruppen har hatt tilgang til og mottatt fra arbeidsgiver, så er det noen få tider gruppen ikke har hatt mulighet til å implementere. Gruppen har derfor kommet til enighet om å bort i fra disse tidene under utviklingen. Dette er mulig å implementere i fremtiden, men som nevnt krever det flere ressurser.

Ved videre arbeid vil det være nødvendig å implementere flere funksjonaliteter for å forbedre brukeropplevelsen. Den første funksjonaliteten er et valg for bruker om å gjøre indikatorene til cellene synlig eller skjult. En annen funksjonalitet er at dersom bruker åpner nedtrekksmenyen til en celle på nedre halvdel av skjermen, hindres denne i å gå utenfor synlig felt på skjermen. Slik at bruker vil se hele nedtrekksmenyen, uavhengig av valgt celles plassering på skjermen. Det vil også være aktuelt å implementere en mulighet for å lagre sitt personlige oppsett, slik at bruker ikke behøver å organisere oppsettet for hver gang applikasjonen lukkes og åpnes igjen. Applikasjonen har heller ikke en funksjonalitet der brukeren har mulighet til å justere på størrelsen på de ulike feltene ved å dra på kantene til feltet. Siden noen av elementene er mer relevant enn andre, så er det viktig å kunne gjøre de mer prioriterte elementene større. Det vil være enklere for brukeren å ha oversikt over de viktigste tidene, samtidig som at andre mindre viktige elementer vises. Dette er ikke implementert grunnet for lite tid til utviklingen.

9 REFERANSER

- Blazor University (2021) JavaScript Interop Tilgjengelig fra https://blazor-university.com/javascript-interop/ (Hentet 06.05.21)
- Cambridge Dictionary (2021) Trial and Error Tilgjengelig fra: https://dictionary.cambridge.org/dictionary/english/trial-and-error (Hentet 10.05.2021)
- Chand Mahesh (2019) What Is WPF. Tilgjengelig fra https://www.c-sharpcorner.com/blogs/what-wpf-is1 (Hentet 26.05.2021).
- GitLab (2021) About GitLab. Tilgjengelig fra: https://about.gitlab.com/ (Hentet 12.04.2021).
- Microsoft (2021) .NET. Tilgjengelig fra: https://dotnet.microsoft.com/ (Hentet 12.04.2021).
- Microsoft (2021) C# documentation. Tilgjengelig fra: https://docs.microsoft.com/en-us/dotnet/csharp/ (Hentet 12.04.2021).
- Microsoft (2021) Blazor. Tilgjengelig fra: https://dotnet.microsoft.com/apps/aspnet/web-apps/blazor (Hentet 12.04.2021)
- Microsoft (2021) Visual Studio. Tilgjengelig fra: https://visualstudio.microsoft.com/ (Hentet 12.04.2021)
- Microsoft (2020) Overview of .NET Framework. Tilgjengelig fra: https://docs.microsoft.com/en-us/dotnet/framework/get-started/overview (Hentet 26.05.2021).
- Microsoft (2014) Data Transfer Object Tilgjengelig fra: https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff649585(v=pandp.10)?redirectedfrom=MSDN (Hentet 10.05.2021)
- Petkar, P. (2020) Agile Workflow. Tilgjengelig fra: https://www.educba.com/agile-workflow/ (Hentet 13.04.2021)
- RedHat (2021) What is a REST API? Tilgjengelig fra: https://www.redhat.com/en/topics/api/what-is-a-rest-api (Hentet 10.05.2021)
- W3C (2016) HTML & CSS. Tilgjengelig fra: https://www.w3.org/standards/webdesign/htmlcss (Hentet 12.04.2021)
- Wålberg, J. A. (2020) Hva er et Gantt-diagram? Tilgjengelig fra: https://www.prosjektbloggen.no/hva-er-et-gantt-diagram (Hentet 13.04.2021)

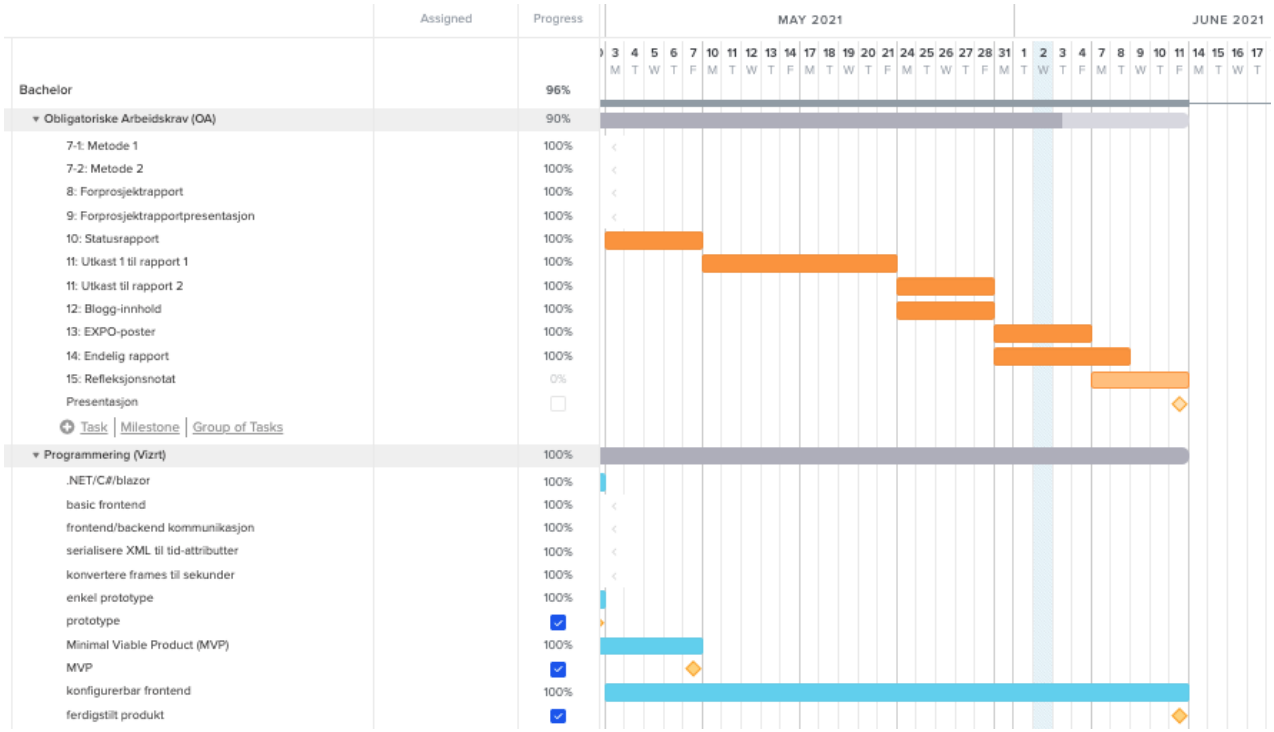
10 APPENDIX

10.1 Risikoliste

Risiko	Sannsynlighet	Konsekvens	Alvorlighet	Håndtering
Covid-19	1	5	5	Følge retningslinjene
Utilstrekkelig kunnskap	3	3	9	Benytte hjelpemateriell på internett.
Prosjektet kan ikke kjøres på gruppens maskiner	3	5	15	Investere i nye maskiner
Brukertesting ikke tilgjengelig	2	4	8	Uten brukertesting, blir det ikke noe konklusjon på om produktet er bra nok.
Arkitektur	3	3	9	Det vil alltid være delte meninger om en arkitektur er bra nok.
Forstyrrelser fra andre fag/eksamen	1	2	2	Gruppemedlemmer må balansere bachelor og andre fag
Misforståelser rundt kravspesifikasjoner	2	3	6	100% klarhet i hvilke resultat som kreves.
Forstyrrelser fra eksterne arbeidsplasser	2	5	10	Jobb må ikke ødelegge for prosjektet.

Figur 10-1: Risikoliste

10.2 GANTT diagram



Figur 10-2: GANTT diagram

10.3 Brukermanual

For å kunne bruke Timing informasjon, trenger du å kjøre Viz Mosart og Mosart Manus Administrator i bakgrunnen. Dette er applikasjoner utviklet av Vizrt, som krever en datamaskin med Windows operativsystem.

10.3.1 Oppstart

1. Kjør Viz Mosart og Mosart Manus Administrator.
2. Kjør Timing informasjon applikasjonen.

10.3.2 Drag & drop

Flytte på celler med informasjon i applikasjonen

1. Hold musepekeren over ønsket celle.
2. Trykk, hold inne og dra cellen til ønsket sted på skjermen.
3. Slipp musepekeren når cellen er innenfor ønsket plassering og innenfor cellens rammer.
4. Cellen med informasjon er nå flyttet.

10.3.3 Nedtrekksmeny

Endre synlig informasjon i valgt celle

1. Holde musepekeren over ønsket celle.
2. Klikk på tannhjulet øverst i høyre hjørne.
3. Klikk på ønsket tid fra nedtrekksmenyen.
4. Tiden i den valgte cellen er nå endret.