Bacheloroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

Alexander Aschlund Jørgensen
Didrik Kielland Bjerk
Kristoffer Haugen
Øyvind Timian Dokk Husveg

Kundeboss

Flyt.cloud

Bacheloroppgave i ingeniørfag, data
Veileder: Frode Haug

Mai 2021

NTNU
Kunnskap for en bedre verden

Alexander Aschlund Jørgensen
Didrik Kielland Bjerk
Kristoffer Haugen
Øyvind Timian Dokk Husveg

Kundeboss

Flyt.cloud

Bacheloroppgave i ingeniørfag, data
Veileder: Frode Haug
Mai 2021

Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

NTNU

Kunnskap for en bedre verden

Sammendrag

Titel:	Kundeboss
Dato:	20.05.2021
Forfattere:	Alexander Aschlund Jørgensen Didrik Kielland Bjerk Kristoffer Haugen Øyvind Timian Dokk Husveg
Veileder:	Frode Haug
Oppdragsgiver:	TietoEvry avdeling Brumunddal
Kontaktpersoner:	Guro Storlien Evensen & Jan Fredrik Gundersen
Nøkkelord:	Programmering, Webutvikling, Scrum, Skyteknologi, CRM-system
Antall sider:	87
Antall vedlegg:	10

Sammendrag: TietoEvry forholder seg daglig til et stort antall kunder, og deres leverandører. Deres nåværende CRM-system er uoversiktlig og informasjonen lagres i flere ulike kanaler. Flere ansatte kan jobbe med samme kunde, og kommunikasjonen mellom partene kan derfor bli vanskelig å få oversikt over. TietoEvry hadde et ønske om å utvikle et nytt system for å løse disse problemene. Gruppen har utviklet et CRM-system som samler all informasjon om kunder og deres leverandører på én plattform. Administrator vil ha full oversikt over alle brukere, og ansatte vil ha tilgang til sine kunder og tilhørende leverandører. Løsningen er implementert med skytjenester fra Microsoft Azure, og er implementert med en serverless arkitektur. Gruppen har forholdt seg til utviklingsprosessen Scrum for gjennomføringen av prosjektet. Med prosjektet har gruppen levert en løsning som kan testes i reelle situasjoner, og senere bli lansert.

Abstract

Title:	Kundeboss
Date:	20.05.2021
Authors:	Alexander Aschlund Jørgensen Didrik Kielland Bjerk Kristoffer Haugen Øyvind Timian Dokk Husveg
Supervisor:	Frode Haug
Employer:	TietoEvry Brumunddal
Contact person:	Guro Storlien Evensen & Jan Fredrik Gundersen
Keywords:	Programming, Web Development, Scrum, Cloud Technology, CRM-system
Pages:	87
Attachments:	10

Abstract: TietoEvry deals with a large number of customers, and their suppliers on a daily basis. Their current CRM system is confusing and the information is stored in several different channels. Several employees can work with the same customer, and the communication between the parties can therefore be difficult to get an overview of. TietoEvry wanted to develop a new system to solve these problems. The group has developed a CRM system that gathers all information about customers and their suppliers on one platform. The administrator will have a full overview of all users, and employees will have access to their customers and associated suppliers. The solution is implemented with cloud services from Microsoft Azure, and is implemented with a serverless architecture. The group has used the Scrum framework during the development of the project. With the project, the group has delivered a solution that can be tested in real situations, and later launched.

Forord

Denne bacheloroppgaven er utarbeidet ved instituttet for datateknologi og informatikk ved NTNU i Gjøvik, og er utført av Alexander Aschlund Jørgensen, Didrik Kielland Bjerk, Kristoffer Haugen, og Øyvind Timian Dokk Husveg.

Gruppen ønsker å takke veileder, Frode Haug ved NTNU, for et kontinuerlig godt samarbeid med ukentlige veiledningsmøter, og god veiledning av gruppen gjennom hele prosjektperioden.

Samtidig ønsker gruppen å takke oppdragsgiver, TietoEvry avdeling Brummundal, spesielt Guro Storlien Evensen og Jan Fredrik Gundersen for mange gode innspill. Gruppen har, så lenge det har latt seg gjennomføre i henhold til anbefalingene vedrørende koronarestriksjonene, tilbragt en dag i uken ved deres kontorer i Brumunddal, og ønsker å takke for denne muligheten. Til slutt ønsker gruppen å takke for en kontinuerlig god oppfølging, og tilgjengelighet gjennom hele prosjektet.

Gruppen ønsker også og takke hverandre for et godt, konstruktivt og lærerikt samarbeid gjennom bacheloroppgaven.

Innhold

Figurer

Kodelister

Akronymer

API Application Programming Interface. viii, 6, 20, 34, 36–40, 44–46, 48, 50, 56, 62, 69–71, 81, 86

Azure AD Azure Active Directory. 10, 36, 39, 41, 66, 102

CDN Content Delivery Network. 34, 45, 66

HTML Hypertext Markup Language. 68

HTTP Hypertext Transfer Protocol. 13

HTTPS Hypertext Transfer Protocol Secure. 13

MQL MongoDB Query Language. 45, 50

MSAL Microsoft authentication library. 51

MSAL.js Microsoft authentication library for JavaScript. 45

NPM Node Package Manager. 45, 58

SLA Service-level Agreement. 66

VSCode Visual Studio Code. 21, 46, 50

XSS Cross-site scripting. 50

Ordliste

access token brukt for å gi brukeren tilgang til ressurser fra API-et. 36, 38, 40, 48, 49, 51, 52

Azure Functions serverless databehandlingtjeneste fra Microsoft. 37, 46, 66

CD kontinuerlig utgivelse. 54, 55

CI kontinuerlig integrasjon/. 54

CI/CD kontinuerlig integrasjon/Kontinuerlig utgivelse. 45

context ett objekt som sendes fra Azure Functions runtime til funksjonen og blir gitt informasjon om funksjonen som blir kjørt, for eksempel statuskode og respons. 70

CRM kunderelasjonhåndtering (fra engelsk Customer Relationship Management). 1, 7

draw.io nettside som tilbyr ett gratis verktøy for dannelse av forskjellige diagrammer. xi, 35, 37–39

FaaS Function as a Service. 37

Git Git blir brukt for å holde syn på forandringer i filer og gjør det enkelt å samarbeide med andre utviklere ved hjelp av forskjellige funksjonaliteter. 82

Jira et oppgavehåndteringssystem utviklet av Atlassian. 16

PaaS Platform as a Service. 81

pattern patterns er mønstre som for eksempel viser til en eksisterende teknikk for utarbeiding av design. 23–25, 82

RESTful representational state transfer, en arkitektur ofte brukt til å lage interaktive applikasjoner med web tjenester. 37

stateless prosessen/serveren holder ikke på informasjon fra tidligere begivenheter. 37

tenant en dedikert og isolert instanse av Azure AD. 36, 66

third-party cookies cookies som er holdt under oppsyn av andre nettsider enn den brukere er på for øyeblikket. . 41

TLS Etterfølgeren til SSL. Det er en kryptografisk protokoll som tilbyr sikker kommunikasjon på Internett. 50

TypeScript supersett av JavaScript. 41

useForm Hook fra React Hook Form biblioteket for kontroll over inputverdier.. 62

Kapittel 1

Introduksjon

1.1 Bakgrunn

TietoEvry er en IT-orientert organisasjon som benytter teknologi for å skape et digitalt fortrinn for deres kunder. Organisasjonen har hovedkontor i Finland, men denne bacheloroppgaven er tildelt gruppen av TietoEvry avdeling Brumunddal[1].

1.2 Problemområde

Større organisasjoner forholder seg ofte til flere kunder. Ved enkelte organisasjoner er det nødvendig å holde kontakt med den aktuelle kundens leverandører, partnere, eller andre aktuelle samarbeidspartnere. Både kunden og samarbeidspartnere kan ha oppgitt flere kontaktpersoner, og samtidig kan flere ansatte jobbe med samme kunde. Som en konsekvens av dette kan kommunikasjon mellom og innad partene oppleves komplisert og innviklet. Dette kan koste organisasjonen tid, penger og ressurser.

1.3 Avgrensning

TietoEvry avdeling Brumunddal forholder seg daglig til flere kunder, og deres tilhørende leverandører. Deres nåværende kommunikasjonssystem medfører at oppbevaring av informasjon, og kommunikasjon med kunder og leverandører oppleves uorganisert og uoversiktlig. Systemet har blitt pekt ut som tidskonsumerende og lite effektivt. Ved å utvikle et CRM-system, som samler kommunikasjon og relevant informasjon, ønsker TietoEvry å effektivisere samhandlingen mellom organisasjonen, kunde, og leverandør.

[1] `https://www.tietoevry.com/no/om-oss/om-tietoevry/`

1.4 Prosjektbeskrivelse

Oppdragsgiver ønsker en webapplikasjon med formål om å samle dialog og relevant informasjon mellom organisasjonen, kunder og leverandører gjennom et oversiktlig brukergrensesnitt. Altså et CRM-system tilpasset deres behov. Under følger en detaljert prosjektbeskrivelse utviklet i samarbeid med oppdragsgiver.

- Webapplikasjonen skal fungere som en portal for organisasjonens ansatte, hvor dialog med kunden og annen relevant informasjon samles ved en plattform. Kunden skal også ha en egen portal. Systemet skal videreformidle relevant informasjon, og delegere tilgang til de ulike brukerrollene.
    - Administrator skal ha tilgang til all data, og vil kunne redigere informasjon ved alle ansatte, kunder og leverandører. Administrator har også mulighet til å administrere rettighetene til brukere av applikasjonen.
    - Organisasjonen skal ha en tilpasset portal for de ansatte. Ansatte skal ha mulighet til å se og endre informasjon om kundene. I tillegg til en integrert mail-funksjonalitet der ansatte kan sende og se eposter mellom organisasjonen, kunder og leverandører.
    - Kundeportalen skal vise alle lagrede data om seg selv, kontaktinfo til leverandør, og aktuell kontaktperson ved TietoEvry. Kundeportalen gir ikke mulighet til å redigere informasjon om seg selv, eller tilgang til mailfunksjonalitet.
- Det skal integreres en mailfunksjonalitet direkte i webapplikasjonen. Det skal være mulig å sende mail til et utvalg relevante mottakere, og samtidig skal det være mulig å svare på mail gjennom webapplikasjonen. Dialog lagres slik at det kan innhentes ved et senere tidspunkt. Det skal også implementeres en metode for å bekrefte at mailen er lest.
- For å gjøre det oversiktlig skal det implementeres stikkord ved kunden. Stikkord vil gjøre det mulig å søke opp alle kunder, og deretter få et oversiktlig bilde av alle kunder ved aktuelt stikkord. Det skal implementeres funksjonalitet for å redigere stikkord ved kunder.
- Brukergrensesnittet skal være enkelt, ryddig, organisert, og utviklet responsivt slik at det er tilgjengelig i et oversiktlig format på ulike enheter.

1.5 Formål

Formålet med prosjektet er å utvikle en forbedring av nåværende praksis omkring oppdragsgivers system vedrørende kundebehandling, organisering og oversiktlighet for å effektivisere deres arbeidsprosess.

1.6 Prosjektmål

Gjennom prosjektet har gruppen definert konkrete og tydelige mål tidlig i planleggingsfasen. Hensikten med utarbeidingen av prosjektmålene var å legge til rette for en målrettet arbeidsprosess mot konkrete mål gjennom hele prosjektperioden. Diverse mål deles inn i effektmål, resultatmål og læringsmål for prosessen, og disse er utarbeidet i samarbeid med oppdragsgiver.

1.6.1 Effektmål

Effektsmålene for oppgaven omhandler effektiviseringen av det gamle systemet, tid, ressurser, og penger ved de involverte partene.

- Redusere tiden en ansatt bruker på å sende, og dokumentere mail med cirka én time for hver behandlede kunde per uke.
    - Etter diskusjon med oppdragsgiver ble det estimert at det er mulig å redusere tidsforbruket med cirka én time per kunde som oversees av en ansatt i løpet av en uke.
- Redusere kostnader ved å legge til rette for en mer effektiv dialog mellom TietoEvry og kunden. En mer oversiktlig dialog mellom organisasjonen og kunden vil redusere muligheten for at viktig informasjon blir glemt og/eller oversett.
    - Redusering av kostnader kan måles opp mot tidsforbruk og antall feil i form av oversett informasjon.
- Redusere tid ved raskere opplæring.
    - Slik systemet er satt opp i dag er det innviklet å skaffe oversikt over hvor ulik informasjon er oppbevart. Ved å redusere opplæringstid vil det frigi ressurser brukt på opplæring.
- Forbedre tilfredshet fra kundene sin side som et resultat av et mer oversiktlig og ryddig system, raskere svar og bedre kommunikasjon, og dermed raskere utførelse av arbeid.
    - Ved å benytte kundeundersøkelser kan det være mulig å dokumentere den graden systemet kan ha en effekt på kundetilfredshet.

1.6.2 Resultatmål

Målet for prosjektet er å utvikle en webapplikasjon som samler informasjon vedrørende TietoEvry sine kunder og leverandører på en plattform. Systemet skal gjøre kunde- og leverandørrelatert informasjon mer strukturert og lettere tilgjengelig gjennom et oversiktlig brukergrensesnitt. Informasjon vil omfatte dialoger mellom kunden og organisasjonen, kontaktdetaljer og annen generell informasjon.

1.6.3 Læringsmål

Læringmålene for prosjektet er definert ved emnebeskrivelsen for bacheloroppgaven[2]. Læringsmålene gir en indikasjon på hvilke læremål som skal oppfylles i løpet av bacheloroppgaven.

Prosjektspesifikt

- Lære å anvende scrum-metodikken i en reel situasjon.
- Lære å planlegge og estimere tid i større prosjekter.
- Lære å samhandle med en oppdragsgiver.
- Lære å dokumentere og henvise til kilder.

Teknologi

- Lære å bruke tester og andre kontrollsystemer.
- Lære å anvende versjonskontroll, ved hjelp av Git.
- Lære å anvende de forskjellige teknologiene, for eksempel skytjenester og databaser.
- Lære å designe brukervennlige og intuitive nettsider.

1.7 Målgruppe

Resultatet av prosjektet vil bestå av et nyutviklet system, og en rapport vedrørende prosjektprosessen og utviklingen av systemet. Målgruppen for resultat vil defineres utifra disse to delene av resultatet.

1.7.1 System

Målgruppen av systemet er oppdragsgiver. Dette inkluderer alle fremtidige brukere av systemet. Hovedsaklig brukere som administrator, ansatte, og deres kunder.

1.7.2 Rapport

Målgruppen for rapporten er alle som ønsker et innblikk i utviklingsprosessen av prosjektet. Dette omfatter medelever, sensor, oppdragsgiver, og eventuelle videreutviklere av systemet.

1.8 Rammer

Rammene ved oppgaven defineres som teknologiske rammer, fysiske rammer, og andre forhold.

[2] `https://www.ntnu.no/studier/emner/BIDAT39#tab=omEmnet`

1.8.1 Fysiske rammer

- Perioden for gjennomførelsen av prosjektet er preget av koronasituasjonen. Det blir stadig innført nye nasjonale tiltak, og disse kan påvirke gruppearbeidet i den grad det ikke lenger vil være mulig å møtes fysisk. Siden det ikke alltid er mulig å organisere fysisk oppmøte har kommunikasjonsplattformer som Microsoft Teams og Zoom blitt alternative plattformer for organiserte møter og diskusjoner.
- Prosjektperioden strekker seg fra 11.01.2021 til 20.05.2021. For å gjennomføre prosjektet slik det er ønsket vil det være viktig å konstant jobbe målrettet mot fristen, uten å havne for mye på etterskudd, gjennom hele perioden.

1.8.2 Teknologiske rammer

- Prosjektoppgaven utvikles slik at den kan benyttes av TietoEvry Brumunddal. Oppdragsgiver ønsker at resultatet tilpasses brukerkapasisteten ved deres egen avdeling, men det skal være mulig å utvide systemet på et senere tidspunkt.
- Det er satt et krav om at oppgaven skal løses med skytjenesteplattformen Microsoft Azure. Grunnen til dette er at oppdragsgiver allerede bruker Azure, og det vil derfor passe inn i deres system.
- Det er informert om anbefalte teknologier og verktøy for gjennomføring, men beslutningen om hvilke andre teknologier og verktøy som benyttes ligger hos gruppen.

1.8.3 Andre føringer

Andre føringer utover tidligere nevnte rammer, defineres i gruppeavtalen. Det er viktig at gruppen gjennom hele prosjektet forbeholder seg til de overordnede lover og regler som er satt. Se side syv og åtte i vedlegg E for utarbeidet gruppeavtale.

1.9 Gruppemedlemmer

Prosjektet er utarbeidet av en utviklingsgruppe bestående av fire studenter, Alexander Aschlund Jørgensen, Didrik Kielland Bjerk, Kristoffer Haugen og Øyvind Timian Dokk Husveg. Alle gruppemedlemmer er studenter ved dataingeniørstudiet ved NTNU i Gjøvik, og utfører prosjektet som avsluttende bacheloroppgave.

1.9.1 Tidligere kunnskap

Gruppen har gjennom studieperioden opparbeidet et godt kunnskapsgrunnlag for å løse bacheloroppgaven. Det faglige utgangspunktet er hovedsaklig likt, men Didrik Bjerk og Alexander Jørgensen har hatt programvaredesignemnet. Didrik har også hatt programvaresikkerhet. Kristoffer Haugen har hatt user centered design

og web-teknologi og vil derfor jobbe med utvikling av frontend. Alle gruppemedlemmer har erfaring fra tidligere utviklingsfag som applikasjonsutvikling, objektorientert programmering og systemutvikling. Gruppen har også innhentet mye informasjon fra andre emner gjennom studieperioden. Denne forkunnskapen gir gruppen et godt utgangspunkt for å utvikle et velfungerende sluttresultat.

1.9.2 Prosjekt- og arbeidsbakgrunn

Gruppen utfører prosjektet uten reell prosjekt- eller arbeidserfaring fra virkelige arbeidsgivere. Tidligere erfaring med prosjekter og problemløsning kommer fra prosjekter i ulike emner hos NTNU. Gruppemedlemene har jobbet sammen ved tidligere prosjekter, og stiller derfor med god kjennskap til hverandre. Gruppen ønsket å benytte denne muligheten til å innhente verdifull erfaring og lærdom gjennom hele prosjektperioden.

1.10 Roller

Under prosjektperioden vil TietoEvry Brumunddal være oppdragsgiver. Frode Haug, ved NTNU, vil fungere som veileder. Innad i prosjektgruppen vil alle gruppemedlemmer fungere som utviklere. Som utvikler har gruppemedlemmer ansvar for å ferdigstille tildelte oppgaver i løpet av aktuell sprint, og de skal også hjelpe resterende gruppemedlemmer om nødvendig. Det vil løpende rulleres på oppgaver som referatskriving, møteinnkalling og oppgavedelegering etter enighet i gruppen.

Gjennom prosjektet vil Kristoffer Haugen være Scrum master. Han har hovedansvaret for å kommunisere med produkteier, oppdragsgiver, og veileder på vegne av gruppen. Han vil ha ansvar for de ukentlige møtene, og vil ha en ekstra stemme ved bestemmelser hvor stemmene ender likt. Han er også utvikler, og vil jobbe mest med frontend.

Alexander Jørgensen vil fungere som prosjektleder og sørge for kontinuerlig fremdrift. Han er ansvarlig for at gruppen når aktuelle delmål til riktig tid. Som utvikler vil han jobbe med frontend.

Didrik Kielland Bjerk fungerer som utvikler. Han jobber med backend, har hatt ansvar for systemdesignet og arbeidet blant annet på databasedesign, API, autentisering og konfigurering av skytjenestene brukt i systemet.

Øyvind Timian Dokk Husveg fungerer også som utvikler. Han jobber med backend, har ansvar for sikkerheten ved blant annet autentisering, og har arbeidet med testing, API og Azure.

Rollene er ikke satt for å begrense arbeidet eller læringen til noen av gruppemedlemmene. Derfor fungerer rollene veiledende og dekker hovedansvaret hos hvert gruppemedlem. Arbeidsoppgaver delegeres etter enighet i gruppen.

1.11 Om rapporten

Rapporten omhandler utviklingsprosessen av et CRM-system for å organisere kommunikasjon og relevant informasjon ved organisasjon, kunde og leverandør. Den inkluderer drøftinger, beslutninger, begrunnelser, og dokumentasjon ved arbeids- og utviklingsprosessen av prosjektet. Rapporten skal beskrive den fulle prosessen for prosjektet, og videreformidle hvordan gruppen har valgt å løse prosjektet på en tilfredstillende måte for alle involverte parter.

Rapporten er bygget opp gjennom følgende ti hovedkapitler.

1. Introduksjon. En introduksjon til problemområdet, en forklaring av hva oppgaven innebærer, og hvordan oppgaven skal løses.
2. Kravspesifikasjon. Et kapittel med fokus på et dypere innblikk i funksjonelle- og ikkefunksjonelle krav for utviklingsprosessen.
3. Utviklingsprosess. Beskrivelse av utviklingsprosessen gjennom prosjektet.
4. Brukergrensesnitt. Et kapittel med fokus på hvilke metoder som er benyttet for valg av design og hvordan vi har implementert disse løsningene.
5. Teknisk Design. Et større bilde av hvilke løsninger som er valgt og begrunnelse for disse løsningene.
6. Implementasjon. Hvilke teknologiske løsninger er implementert og hvordan dette er utført. Kapittelet drøfter også hvorfor disse løsningene er valgt for utførelse av prosjektet.
7. Utrulling. Beskrivelse av hvordan systemet kan skaleres etter ønske og hvordan bruker kan administrere systemet.
8. Testing og kvalitetssikring. Beskrivelse av testingen som er utført gjennom utviklingsprosessen, og hvordan dette har påvirket sluttresultatet.
9. Diskusjon. Et drøftingskapittel om utførelsen av prosjektet fra gruppens perspektiv.
10. Konklusjon. Konklusjon av prosjektet og hva som er oppnådd gjennom prosjektperioden.

Kapittel 2

Kravspesifikasjon

Kravspesifikasjon av tildelt oppgave er utarbeidet i forkant av utviklingsprosessen for å kommunisere og utarbeide de eksakte krav og spesifikasjoner oppdragsgiver forventer av oppgaveløsningen.

2.1 Use case

Ved å drøfte problemråde og oppgavebeskrivelse med oppdragsgiver har gruppen utarbeidet lavnivå og høynivå use caser for det tiltenkte systemet. Dette gir et konkret bilde av det som kreves av oppdragsgiver for en tilfredstillende løsning av oppgaven.

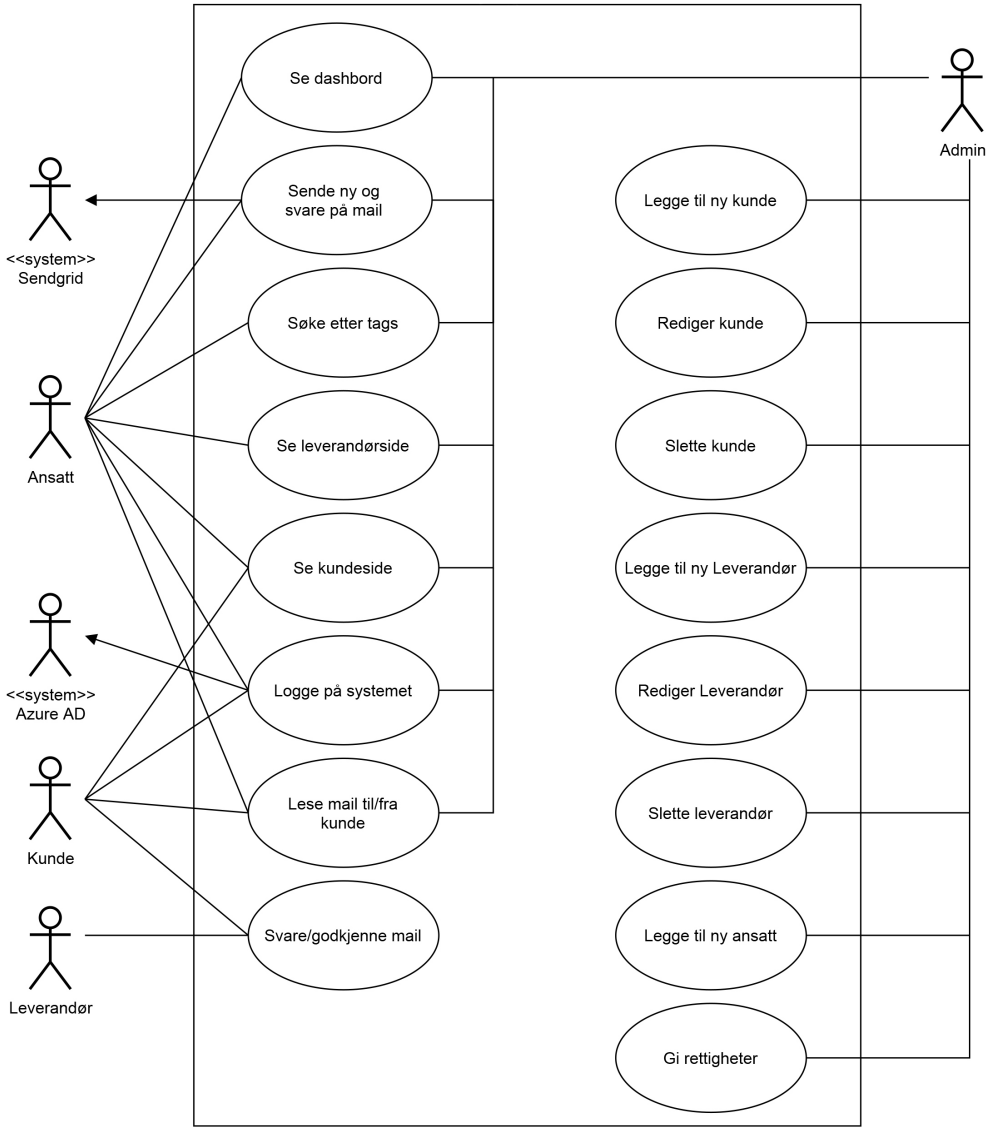
De utarbeidede use casene har gjennom hele prosjektet gjort det enkelt å holde oversikt over hvilke funksjonaliteter det ferdigstilte systemet skal tilby. For å sikre at alle use case ferdigstilles til riktig tid, har gruppen gjennom hele prosjektet basert sprintoppgaver i backlog etter blant annet disse use casene.

2.2 Høynivå use case

Høynivå use case gir et overordnet bilde av de ulike funksjonalitetene ved systemet, og demonstrerer tilgangen til aktørene.

2.2.1 Use case diagram

Ved utarbeidingen av use casene konstruerte gruppen et use case diagram. Diagrammet gir et oversiktlig bilde av funksjonalitet og hvilke tilganger de ulike brukergruppene har, samt rollene til eksterne aktører ved systemet. Videre har dette blitt brukt til å definere en product backlog for planlegging av utviklingsprosessen.



Figur 2.1: Use case diagram for applikasjonen

2.2.2 Use case

Videre har gruppen definert de samme funksjonalitetene ved use caser. Nedenfor følger et utdrag av de utarbeidede høynivå use casene, men det fulle utvalget er vedlagt i vedlegg C.

Navn: Legge til ny kunde.
Aktør: Admin.
Mål: Lage en ny kunde, og fylle inn informasjon.
Beskrivelse: Trykker på "ny-kunde"-knapp. Den blir da tatt til en side der den kan legge inn informasjon om kunden. Informasjonen som må legges inn er navn og kontaktperson. Kan legges til kommentar, tags, kunder og referanse.

Navn: Legge til leverandør på kunden
Aktør: Admin og ansatt
Mål: Opprette en relasjon mellom kunden og leverandør
Beskrivelse: Bruker går inn på en kundeside og trykker på "legg til leverandør"-knappen. Den kan da velge en eksisterende leverandør og opprette en relasjon mellom kunden og leverandøren.

Navn: Lese mail til/fra kunde
Aktør: Admin og ansatt
Mål: Lese mailhistorikken ved en kunde
Beskrivelse: Bruker med tilgang til kunden kan se mailhistorikken ved kunden. Den skal også kunne se mailene til og fra de tilknyttede leverandørene.

Navn: Autorisering / Innlogging
Aktør: Admin, ansatt og kunde
Mål: Aktøren logger seg på applikasjonen
Beskrivelse: Aktørene kommer til et startvindu der de kan logge seg på ved hjelp av Azure Active Directory (Azure AD). Aktørene logger på og blir tatt til sin startside:

- Kunden sendes til sin kundesiden.
- Kunden blir sendt til dashbordet med oversikt over kunder.
- Admin blir sendt til en startside hvor den kan velge å se administatorrettigheter, eller dens kunder.

Navn: Gi rettigheter til ny ansatt
Aktør: Admin
Mål: Gi rettigheter til ny ansatt
Beskrivelse: Admin kan navigere ansatte ved ansattsiden og delegere rettigheter til ansatte.

2.3 Lavnivå use case

Enkelte av use casene kan fremstå mer krevende og kompliserte å implementere, og derfor kan en mer detaljert beskrivelse av use caset gi et klarere bilde av hva som skal implementeres. Under følger et utvidet use case hvor ansattes mailfunksjonalitet blir beskrevet.

Navn: Ansatte sender mail til en kunde.
Mål: En ansatt kan sende mail til en valgt kunde gjennom applikasjonens mailfunksjonalitet.
Pre-Betingelse: En ansatt er logget inn med internettilkobling og befinner seg på dashbordet for utførelse i samsvar med beskrevet programflyt nedenfor.
Post-Betingelse: Ansatt har navigert seg til mailfunksjonalitet og fylt ut informasjonsfelt før mail kan sendes.
Beskrivelse: Fra dashbordet kan den ansatte navigere seg til den aktuelle kunden for å svare på et eksisterende emne eller opprette en ny mail. Det er også mulig å opprette et nytt emne fra mailknappen på dashbordet.

---

1. Den ansatte benytter seg av mailfunksjonalitet på dashbordet og trykker på denne knappen.
    a. Aktør befinner seg ved dashbord.
    b. Aktør søker i kunden ved navn eller stikkord for å finne korrekt kunde. Kunder med tilsvarende stikkord eller navn vises vises.
    c. Aktør klikke på aktuell kunde og blir tatt til kundesiden.
    d. Aktør klikker på sendmail knapp ved kundesiden.
    e. Aktør legger til mottakere. Aktuell kunde er standard mottaker, men kan fjernes. Andre aktuelle mottakere er kunder og leverandører tilhørende aktøren som ønsker å sende mail.
    f. Aktør skrive emne og innhold.
    g. Aktør trykker på sendknapp.
    h. Mailen sendes til valgte mottakerne, og en kopi blir lagret i databasen.

    eller

    a. Aktør befinner seg ved dashbord.
    b. Aktør markerer aktuell kunde og trykke på send mail.
    c. Aktør legger til mottakere. Aktuell kunde er standard mottaker, men kan fjernes. Andre aktuelle mottakere er kunder og leverandører tilhørende aktøren som ønsker å sende mail.
    d. Aktør skrive emne og innhold.
    e. Aktør trykker på sendknapp.
    f. Mailen sendes til valgte mottakerne, og en kopi blir lagret i databasen.

Feil:
Ingen treff på navn eller stikkord.

1. Ansatt får ikke opp noen kunder som passer med søket.
2. Ansatt kan endre søket slik at det passer, eller lete manuelt gjennom listen.

Mailen mangler emne, innhold eller mottaker.

1. Ansatt trykker på send før alle informasjonsfelt er utfylt.
2. Det blir informert om manglende utfylt informasjonsfelt.
3. Ansatt går tilbake og fyller ut manglende informasjon før den kan sendes.

2.4 Operasjonelle krav

De operasjonelle kravene er utviklet i samarbeid med oppdragsgiver, TietoEvry avdeling Brumunddal.

1. Systemet skal kunne ha 100 samtidige brukere.
    - Systemet er et pilotprosjekt og skal brukes av TietoEvry avdeling Brumunddal og deres kunder. I startfasen er det ikke forventet at flere enn 50 personer bruker webapplikasjonen systemet samtidig. Systemet skal utvikles slik at det er mulig å skalere til større bruk ved en senere anledning.
2. Oppdragsgiver krever at systemet skal ha en oppetid på minst 99.9% i arbeidstiden. Det vil si klokken 08:00 til 16:00 på hverdager. Dette vil gi en nedetid på cirka tre og ett halvt minutt per uke.
3. Maksimal svartid fra server skal ikke overstige tre sekunder og gjennomsnittstiden skal ikke overstige ett sekund.

Utarbeidingen av de operasjonelle kravene er diskutert ved senere anledninger i rapporten. Henholdsvis ved følgende kapitler.

- Punkt 1 diskuteres ved kapittel 5.2.1.
- Punkt 2 diskuteres ved kapittel 7.2.
- Punkt 3 diskuteres ved kapittel 5.2.2.

2.5 Sikkerhetskrav

Sikkerhetskravene omhandler brukertilganger og oppbevaring av sensitiv informasjon. I samarbeid med oppdragsgiver har gruppen utarbeidet diverse sikkerhetskrav for systemet.

1. Alle brukere skal bli autentisert før de får tilgang til applikasjonen.
2. Det skal implementeres autorisering av roller for å begrense brukernes tilgang.

    - Bare administratorrollen skal ha full tilgang til alle brukere.
    - TietoEvry sine ansatte skal kun se sine tildelte kunder.
    - Kunden skal kun se informasjon om seg selv.
3. Bruke Hypertext Transfer Protocol Secure (HTTPS) for å kryptere Hypertext Transfer Protocol (HTTP)-forespørsler.

Disse sikkerhetskravene har gjennom utviklingsprosessen stått tydelig ved hvilke løsnings- og implementeringsmetoder som har blitt utført. Hvordan dette blir implementert diskuteres senere i rapporten.

- Punkt 1 diskuteres ved kapittel 5.4 og 6.3.
- Punkt 2 diskuteres ved 6.3.1.
- Punkt 3 diskuteres om backend i 6.2.2, frontend i 7.1.1 og database i 6.5

Kapittel 3

Utviklingsprosess

Prosjektet stiller krav til opprettholdelse av tidsfrister, og et resultat som tilfredsstiller kravene som spesifiseres ved kravspesifikasjonen, se kapittel 2. Ved valg av utviklingsmodell vil gruppen ta hensyn til diverse kriterier, og valget vil styre utviklingsprosessen for prosjektet.

3.1 Valg av utviklingsmodell

Valg av utviklingsmodell ble diskutert i oppstartsfasen av prosjektplanleggingen. Gruppen bestemte at Scrum ville passe godt på bakgrunn av følgende kriterier og drøftinger.

3.1.1 Kriterier

Etter diskusjoner vedrørende prosjektgjennomføring har gruppen prioritert å vektlegge følgende kriterier ved valg av utviklingsmodell.

1. Ha rom for nye ideer og endringer underveis.
2. Prosjektet har en konkret tidsfrist uten slingringsmonn.
3. Det er nødvendig å teste produktet før ferdigstillelse.
4. Oppdragsgiver vil følge utviklingsprosessen.

Disse kriteriene er hovedsaklig avgrensninger ved prosjektperioden som gruppen er nødt til å følge. Det er derfor helt essensielt å legge til rette for at disse kriteriene blir fulgt på en best mulig måte.

De spesifiserte kriteriene peker mot en smidig utviklingsmodell hvor det er mulig å gjennomføre justeringer underveis. Det burde også være mulig å sette konkrete tidsfrister ved utarbeiding av delmål.

3.1.2 Drøfting av utviklingsmodell

Gjennom drøftingen av utviklingsmodell brukte gruppen tid på å diskutere flere modeller for å finne den som virket best tilpasset prosjektsituasjonen. Gruppen dannet et overordnet bilde av flere utviklingsmodeller for å sikre riktig valg av modell.

Ved oppstart av prosjektet fikk gruppen en innføring i hvilke tanker oppdragsgiver hadde for det ferdigstilte systemet, hovedsakelig omkring hvilke funksjonaliteter løsningen burde omfatte. Oppdragsgiver gjorde det klart at gruppen oppfordres til å bidra med innspill for utvidelser og/eller justeringer ved systemet slik at krav og spesifikasjoner kan modifiseres under utviklingsprosessen. Samtidig kan også oppdragsgiver justere sine krav gjennom prosjektperioden.

I 2012 publiserte S. Balaji og Dr. M. Sundararajan Murugaiyan en metaanalyse hvor de drøftet fordeler og ulemper ved smidig-, fossefalls- og V-modellmetodikk[1]. Konklusjonen av denne metaanalysen var at beste tilpassede utviklingsmodell varierer utifra aktuell prosjektsituasjon, og at en prosjektsituasjon hvor korte tidsfrister og en justerbar kravspesifikasjon definerer utviklingsprosessen vil smidige-utviklingsmodeller være en foretrukkende løsning. Det ble konkludert med at V-modell var bedre tilrettelagt lengre prosjekter, og fossefallsmetoden la ikke godt nok tilrette for endringer i spesifikasjoner og krav underveis i utførelsen.

Det finnes allerede et stort antall smidige utviklingsmetodikker, men gruppen hadde i hovedsak to prinsipper for valget av modell. Det burde være tilpasset gruppens prosjektsituasjon, og ikke fremstå for komplisert å implementere. Ved metaanalysen *Agile software development methods: Review and analysis* fra 2017 blir smidige utviklingsmetodikker diskutert opp mot hverandre for deres fordeler og ulemper[2]. Metaanalysen definerer utviklingsmodeller som Scrum til å være best tilpasset mindre utviklingsgrupper, som passer gruppens prosjektsituasjon[2]. Ettersom gruppen har blitt anbefalt Scrum utviklingsmetodikken av oppdragsgiver, og samtidig har erfaring med Scrum fra tidligere utviklingsprosjekter har gruppen valgt å benytte Scrum for utviklingsprosessen.

Konklusjonen vedrørende drøftingen av utviklingsmodell var dermed at utviklingsmodellen Scrum ville passe prosjektsituasjonen best. Scrum gir rom for at endringer i kravspesifikasjonen kan oppstå underveis, og samtidig vil regelmessige sprinter gjøre det mulig å opprettholde konkrete frister for delmål[1]. Gruppen og oppgavegiver har god kjennskap til Scrum, og begge parter tror dette vil passe prosjektsituasjonen godt.

3.1.3 Drøfting av sprintlengde

Ettersom gruppen har valgt utviklingsmodellen Scrum for utviklingsprosessen, er gruppen nødt til å definere en sprintlengde for sprinter ved prosjektetutførelsen. Det finnes mye dokumentasjon vedrørende valg av sprintlengde, og gruppen har vurdert valget av sprintlengde mot eksisterende dokumentasjon.

En artikkel utarbeidet av Dan Rawsthorne, en sertifisert Scrum-trener, definerer ulike forbehold gruppen burde ta hensyn til før valg av sprintlengde[1][3]. Rawsthorne definerer at sprintlengden skal være konstant gjennom hele utviklingsprosessen, og beskriver den ideele lengden som et sted mellom ”for lang” og ”for kort”[3]. En for lang sprintperiode vil ikke i like stor grad ta hensyn til justeringer av spesifikasjoner, og en for kort sprintperiode vil gjøre det vanskeligere å ta hensyn til at planlagte sprintoppgaver blir gjennomført[3]. Fra gruppens perspektiv vil det være viktig å utelukke for lange sprinter da spesifikasjoner og krav ved prosjektet kan endres hyppig. Gruppen ønske å utarbeide mindre oppgaver for minimering av risiko vedrørende gjennomføring av sprintmål, og vil ikke delegere for store deloppgaver som ikke lar seg gjennomføre i løpet av en sprintperiode. Hovedargumentet Rawsthorne trekker frem er at sprintlengden burde vurderes utifra den aktuelle prosjektsituasjonen[3].

Overordnet kan gruppens prosjektsituasjon beskrives som kortsiktig, og med et mindre antall gruppemedlemmer. Utifra tidligere erfaring har gruppen lagt vekt på at sprintperiodene skal være store nok til å utarbeide aktuelle oppgaver, men samtidig har gruppen lagt stor vekt på at de er korte nok til at en eventuell dårlig gjennomført sprint ikke vil få for stor effekt for utviklingsprosessen.

Samtidig som korte sprinter kan bidra til mindre risiko knyttet mot sprintene, tar det også mer tid til planleggingsarbeid og gjennomføring av møter. Gruppen har sett på dette som en positiv ting for å ha en kontinuerlig god oversikt over den fulle prosessen.

For å konkludere ønsker gruppen å gjennomføre hyppige sprinter for redusert risiko ved ukontrollerbare hendelser som kan gjøre det vanskelig å fullføre aktuell sprint. Etter diskusjoner innad i gruppen har det blitt enighet om å utføre én ukes sprintperioder. Ved denne sprintlengden vil det ligge mindre risiko ved en dårlig gjennomført sprint, og dersom gruppen hadde valgt lengre sprintperioder vil dette ramme prosjektprosessen i en større grad ettersom prosjektperioden kun strekker seg over en kortere tidsperiode. Gruppen ser det sannsynlig at det kan oppstå problemer ved gjennomføring av en sprint ved for eksempel sykdom eller dårlig estimering av arbeidsmengde. Risikoanalysen for prosjektet er vedlagt ved E. Samtidig vil justeringer i krav og/eller spesifikasjoner kunne kreve rask endring ved prosjektfremgangsmåten, og en kort sprintperiode legger til rette for at dette kan vurderes raskt.

3.2 Gjennomføring

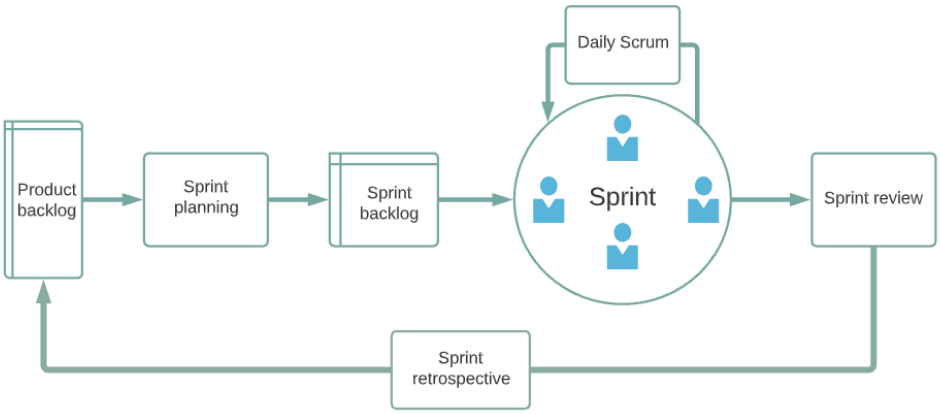
Scrum er, som beskrevet ved 3.1.2, en smidig utviklingsmodell med fokus på å gjennomføre planlagte delmål i sprintperioder. Gruppen organiserte sprinter gjennom hele perioden. Jira ble benyttet som verktøy for sprintorganisering med et *Scrum board* og en *product backlog*. Gruppen hadde regelmessige Scrum-møter

[1] `https://www.scrumalliance.org/community/profile/drawsthorne`

for å organisere arbeidet. Samtidig ble det satt opp møter med oppdragsgiver, veileder, og innad i gruppen der det viste seg å være nødvendig.

3.2.1 Utførelse av Scrum

Ved valgt utviklingsmodell benyttet gruppen seg av de definerte elementene i Scrum-metodikken for korrekt gjennomføring i henhold til modellen[2]. Gruppen utarbeidet en *product backlog*, og har organisert regelmessige sprintmøter med planlegging av sprint backlog, og evalueringer av sprintene. I tilleg til daglige Scrum-møter når gruppen jobbet sammen, se 3.2.4. Figuren 3.1 demonstrerer utførelsen av en sprint i henhold til gruppens utførelse.



Figur 3.1: Scrum framework (laget i draw.io)

Ved oppstartsfasen av prosjektet utarbeidet gruppen, som nevnt, en product backlog. Ved å se på hovedelementene ved kravspesifikasjonen, use casene, og rapportrelaterte deloppgaver, definerte gruppen deloppgavene for utarbeiding av hele prosjektet. Dersom det etterhvert skulle skje endringer ved kravene ble product backlog oppdatert etter de nye endringene underveis i prosjektperioden.

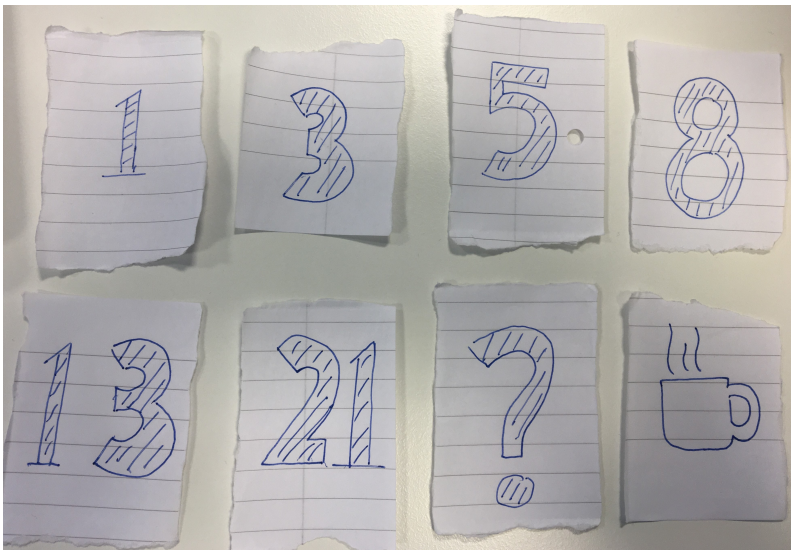
3.2.2 Sprintevaluering

Ved hvert Scrum-møte startet gruppen med en diskusjon av forrige sprint. Sprintevalueringen dreide seg hovedsaklig om sprintmålet og deloppgavene ble gjennomført, og om hvordan sprinten hadde fungert for hvert gruppemedlem. Det ble diskutert hvordan arbeidsinnsatsen hadde vært, hva som hadde gått bra, hva som kunne bli gjort bedre, og hvordan estimeringen av forrige sprintplanlegging fungerte i praksis.

3.2.3 Sprintplanlegging

Sprintplanlegging for kommende sprint diskuteres ved hvert Scrum-møte hvor gruppen starter med å definere et mål for sprintperioden i henhold til utarbeidet

Gantt-diagram og aktuelle delmål, se vedlegg E. Gruppen plukker dermed ut deloppgaver fra product backlog som skal gjennomføres i henhold til de spesifiserte målene. Dersom det er nødvendig å opprette nye oppgaver for gjennomførelse av sprintmålet blir dette utarbeidet ved planleggingen.



Figur 3.2: Planning poker lapper

Ved sprintplanleggingen estimeres arbeidsmengde ved oppgavene slik at gruppen unngår for mye eller lite arbeid gjennom sprinten. Ved estimeringen benyttet gruppen seg av estimeringsmetoden planning poker, hvor hvert gruppemedlem selv estimerer arbeidsmengden i timer for deretter diskutere og komme til enighet med resten av gruppemedlemmene. Estimeringsmetodikken ved bruk av planning poker har gjennom dokumenterte studier vist seg å være svært presist i forhold til både ekspertestimering og det fakktiske resultatet[4].

Ved gjennomføringen av planing poker har gruppen benyttet Fibonaccitallene som story points. Fibonaccitallene øker i samsvar med usikkerheten lagt ved estimeringen, og derfor fungerer tallrekken som en god måte å estimere mindre og større oppgaver[5]. Story point to er utelatt ettersom gruppen har sett det unødvendig ettersom man kan velge en og tre. Gruppen produserte selv lapper etter fibonaccitallene, fra 1 til 21, hvor story points representerte arbeidstimer, se 3.2. Gruppen har satt et maks antall story points til 21 ettersom større oppgaver enn dette burde deles opp i flere mindre deloppgaver.

3.2.4 Daglige Scrum-møter

Gruppen jobbet hovedsaklig med bacheloroppgaven hver eneste dag, og arrangerte enten videomøter, fysisk oppmøte på skolen, eller hos oppdragsgiver. Ved disse møtene ble det kort diskutert hvilke oppgaver gruppemedlemene arbeidet med,

og hvor langt de hadde kommet i arbeidet. Disse daglige Scrum-møtene holdt gruppen kontinuerlig oppdaterte vedrørende sprintprosessen.

3.2.5 Møter med oppdragsgiver

Gjennom Guro Storlien Evensen og Jan Fredrik Gundersen, begge ved TietoEvry avdeling Brumunddal, ble det arrangert møter etter behov gjennom hele prosjektperioden. Gruppen møtte hos TietoEvry Brumunddal, hver onsdag det passet seg i henhold til anbefalingene rundt koronasituasjonen. Det ble også opprettet en kanal i Microsoft Teams hvor vi kunne holde kontakt og arrangere møter. Det ble satt av tid til ukentlige hver onsdag dersom noen av partene følte behov for dette. For referater fra møtene, se vedlegg G.

3.2.6 Møter med veileder

Gruppen avtalte møter med veileder Frode Haug, ved NTNU, hver tirsdag gjennom hele prosjektperioden. Møtene med veileder dreide seg hovedsaklig om den generelle prosessen for bacheloroppgaven og rapportskriving. Møtene ville utgå dersom gruppen ikke hadde noe å ta opp. I begynnelsen av semesteret omhandlet møtene den generelle prosessen om bacheloroppgaven veiledning om tidsfrister, og hvordan ting burde utføres. Mot slutten omhandlet møtene hovedsaklig ferdigstillingen av rapport. Møtereferater med veileder ligger i vedlegg G.

3.3 Sprintoversikt

Sprintoversikten viser hovedpunktene i sprintene. Denne oversikten er utarbeidet etter referat fra Scrum-møte, se vedlegg G, og oversikten på Jira. For mer utfyllende informasjon om sprintene, se vedlegg G.

Sprint: Forprosjekt
Dato: 12.01-31.01
Starte prosjektet og utvikle en prosjektplanen og forprosjektsrapport. Møtepan for oppdragsgiver og veileder ble også bestemt.

Sprint: 1
Dato: 03.02-10.02
Utarbeidet user stories og use caser. Satt opp database, enkel nettside og react-applikasjon.

Sprint: 2
Dato: 10.02-17.02
Videre arbeid med user stories. Sikkerhet- og operasjonelle-krav samt database-modellering og design av nettsiden er påbegynt.

Sprint: 3
Dato: 17.02-24.02
Det ble undersøkt rund sikkerhet og funksjonalitet i backend. Nettsiden ble koblet sammen med backend.

Sprint: 4
Dato: 24.02-03.03
Navn og logo ble laget. Frontend ble refaktorert og flere funksjonaliteter ble lagt til i backend.

Sprint: 5
Dato: 03.03-10.03
Dokumentert backend-funksjoner, undersøkt mail funksjoner og definerte kode-standarder. har også prøvde å implementere testing med varierende suksess,

Sprint: 6
Dato: 10.03-17.03
Flere sider ble implementert i frontend. Lagt til mail-funksjonalitet i backend i tilleg til annen funksjonalitet.

Sprint: 7
Dato: 17.03-24.03
Mer funksjonalitet ble lagt til i frontend og backend.

Sprint: 8
Dato: 24.03-07.04
Statusmøte med oppdragsgiver for å oppklare uklarheter. API ble dokumentert i tillegg til implementasjon av mer funksjonalitet.
I påsken skrev gruppemedlemmene et kapittel hver i den endelige rapporten.

Sprint: 9
Dato: 07.04-14.04
Videreutvikle frontend. SSL sertifikat er lagt til på nettsiden. Og gruppen har jobbet med rapporten.

Sprint: 10
Dato: 14.04-21.04
Gruppen jobbet med rapporten.

Sprint: 11
Dato: 21.04-28.04
Skrive ferdig utkast til rapporten. Nettsiden ble redesignet.

Sprint: 12
Dato: 28.04-05.05
Laget unit-, integrasjon- og brukertester. Mer av nettsiden ble redesignet. Fortsette med rapporten.

Sprint: 13
Dato: 05.05-12.05
Rette på feil og småjusteringer etter brukertest. Rapportskriving. Siste redesign av siden.

Sprint: 14
Dato: 12.05-19.05
Fikse det siste på nettsiden, skrive ferdig rapport og levere.

3.4 Verktøy for utviklingsprosessen

3.4.1 Jira

Som nevnt ved kravspesifikasjonen, 2.1, har gruppen benyttet seg av prosjektstyringsprogramvaren Jira. Jira er utviklet av Atlassian, og blir brukt til for å styre prosjekter med smidige utviklingsmetoder[2]. I løpet av prosjektet ble Jira Scrum board og backlog-funksjonaliteten brukt. Den holder oversikt over oppgaver med story points, og hvem som har ansvaret for oppgavene. Gruppens progresjon, og alle gjennomførte deloppgaver i Jira beskrives ved vedlegg J.

3.4.2 Clockify

Clockify er en applikasjon som gir oversikt over hvordan arbeidsprosessen gjennom prosjektet ser ut[3]. Gruppen har benyttet Clockify for å dokumentere arbeidsmengde og arbeidstema i timer hver dag gjennom hele prosjektet. Timelogg for gruppen er vedlagt ved vedlegg I.

3.4.3 Visual Studio Code

Visual Studio Code (VSCode) er en kodeeditor utviklet av Microsoft[4]. VSCode er enkel å modifisere slik at brukeren får en god tilpasset opplevelse. For eksempel så har utviklerene installert forskjellige programvareutvidelser som en Azure Functions- og prettify-addon.

[2] `https://www.atlassian.com/software/jira`
[3] `https://clockify.me`
[4] `https://code.visualstudio.com`

3.4.4 Figma

Figma er en applikasjon som gjøre det mulig å utarbeide prototyper for hvordan systemet skal se ut[5]. Gjennom Figma har gruppen designet prototyper av både funksjonalitet og komponenter som senere har blitt implementert i prosjektet. Se 4.1.2 for nærmere beskrivelse av gruppens bruk av Figma.

3.4.5 GitHub

Gruppen har benytet GitHub for lagring og deling av kode[6]. GitHub har også fungert som bindeledd mellom publisering av kode, og utgivelser av systemet. Se 7.1 for beskrivelse av gruppens bruk av Github.

3.4.6 Microsoft Teams

Microsoft Teams er en kommunikasjonsplattform utviklet av Microsoft[7]. Gruppen har brukt plattformen for å ha videosamtaler innad i gruppen og med oppdragsgiver når det fysisk oppmøte har vært vanskelig å gjennomføre. Plattformen har også blitt brukt til å dele dokumenter og kommunikasjon gjennom meldinger. Oppdragsgiveren oppfordret også til bruk av Microsoft Teams ettersom de er godt kjent med plattformen.

3.4.7 Zoom

Zoom er, på samme måte som Microsoft Teams, en plattform for videosamtaler, og har blitt benyttet for møter med veileder gjennom hele prosjektperioden[8].

3.4.8 Facebook Messenger

Facebook Messenger er en plattform utviklet av Facebook for å sende meldinger. Gruppen benyttet denne plattformen for å kommunisere utenfor arbeidstid og blant annet planlegge møter og stille spørsmål[9].

[5] https://www.figma.com/
[6] https://github.com/about
[7] https://support.microsoft.com/en-us/office/welcome-to-microsoft-teams-b98d533f-118e-4bae-bf44-3df2470c2b
[8] https://zoom.us
[9] https://www.messenger.com/features

Kapittel 4

Grafisk design

Brukergrensesnitt- og designkapittelet omhandler den funksjonelle og estetiske utviklingsprosessen, og medfølgende hensyn fra utviklingen gjennom hele prosjektperioden. Brukergrensesnittet er nøye utarbeidet for å optimalisere brukeropplevelser ved et enkel og oversiktlig utseende uten for mye distraksjoner. Gjennom kontinuerlig dialog med oppdragsgiver var det mulig å tilpasse justeringer ved utarbeidede prototyper allerede tidlig i utviklingsprosessen. Kontinuerlig dialog med oppdragsgiver gjorde det mulig å tilpasse endringer ved prototypemodellering tidlig i utviklingsprosessen. Samtidig som oppdragsgiver stiller krav til brukergrensesnitt og design, gjør også gruppen det. Etter diskusjoner har gruppen utarbeidet prosjektet med hensyn til valgte patterns, designprinsipper, og rådene for universell utforming [6].

4.1 Utviklingsprosess

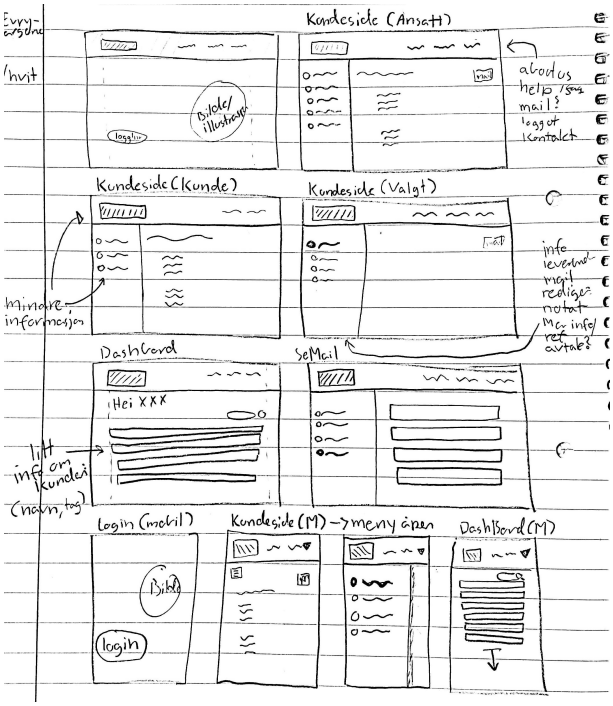
Gjennom utviklingsprosessen har gruppen måttet forholde seg til ulike hensyn, og samtidig ta viktige valg for hvordan resultatet av sluttproduktet skulle utarbeides. Gruppen har konstruert prototyper av ulike nivåer gjennom hele prosessen for å sikre at de implementerte funksjonalitetene og komponentene oppfyller de kravene gruppen har definert for patterns, designprinsipper, og rådene for universell utforming[6]. Samtidig drøftes alltid større avgjørelser med oppdragsgiver.

4.1.1 Low fidelity prototyping

Ved oppstartsfasen av prosjektet brukte gruppen low fidelity prototyper for å sikre at gruppemedlemene hadde lik overordnet oppfatning av hvordan oppgaven skulle utføres[1]. Denne prototypingen i oppstartsfasen av prosjektet gjorde det også mulig for oppdragsgiver å peke ut misforståelser, endringer, og justeringer ved

[1] https://blog.adobe.com/en/publish/2017/11/29/prototyping-difference-low-fidelity-high-fidelity-prototypes
html#gs.1faaai

gruppens forståelse av oppgaven. Figur, 4.1, er en av gruppens første low fidelity modeller for systemet.



Figur 4.1: Low fidelity prototype.

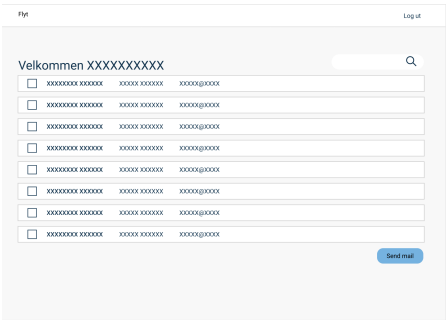
4.1.2 High fidelity prototyping

Gruppen benyttet Figma til å konstruere high fidelity prototyper for systemet gjennom utviklingsfasen av prosjektet. Målet med high fidelity prototypen er å demonstrere hvordan det ferdigstilte brukergrensesnittet vil fungere og se ut. Dette vil si at prototypen inkluderer brukerinteraksjoner, og design i tråd med de kravene gruppen har satt til prosjektet gjennom patterns, prinsipper, og rådene for universell utforming. Figur, 4.2, viser til en demonstrasjon over hvordan utarbeidet prototype ser ut i Figma.

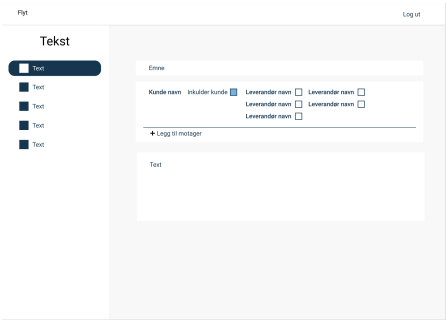
4.1.3 Møter og brukertesting

Gjennom prosjektet har gruppen organisert møter hvor gruppemedlemene sammen har diskutert løsninger for aktuelle problemområder og implementasjoner angående brukergrensesnitt og design. Etter at gruppen har kommet frem til en løsning har denne blitt presentert for oppdragsgiver som deretter gir sine innspill og/eller tanker om løsningen, se møtereferater ved vedlegg G.

Mot slutten av prosjektet utførte gruppen brukertesting for å avdekke potensielle forbedringer eller misforståelser ved systemet. Som en konsekvens av koronasi-



(a) Prtotype dashbord



(b) Prototype mail

Figur 4.2: Eksempel på high fidelity prototype ved Figma

tuasjonen ble utvalget av deltakere begrenset til medstudenter og familie. Dette ble gjennomført for å sikre et godt, oversiktlig og intuitivt brukergrensesnitt. Detaljert fremgangsmåte for brukertestingen er dokumentert i kapittelet om testing 8.3.

4.2 Patterns

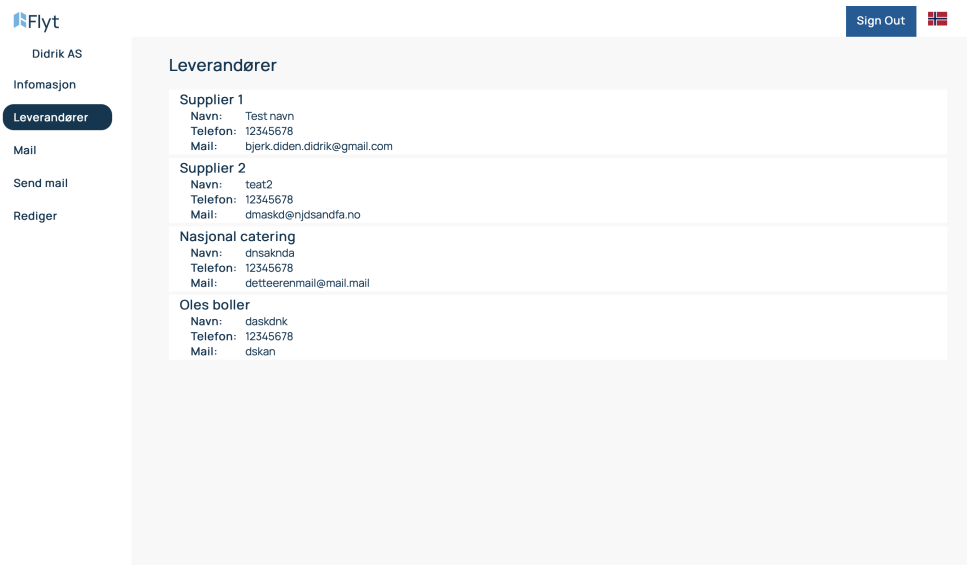
Funksjonalitet og design er i stor grad repetitivt, og patterns gjør det mulig å bruke allerede eksisterende løsninger for diverse designproblemstillinger. Ved å bruke patterns kan gruppen gi en faglig dokumentert begrunnelse for designvalgene som blir tatt gjennom utviklingsprosessen.

4.2.1 Modulfaner

Modulfaner er et pattern brukt for å unngå unødvendig navigering for å innhente informasjon[2]. Gruppen har benyttet seg av dette patternet fra tidlig i oppstartsfasen, og hovedprinsippet baserer seg på smidig navigering mellom informasjon uten behov for å bevege seg frem og tilbake for informasjonen. Ved å implementere en sidemeny kan brukeren enkelt navigere mellom de ulike informasjonssidene

[2] `http://ui-patterns.com/patterns/ModuleTabs`

ved en kunde eller leverandør.



Figur 4.3: Ferdigstilt sidemeny.

Ved 4.3 blir implementasjonen av modulfaner demonstrert. Sidemenyen kan enkelt navigeres, og den aktuelle siden blir er markert ved menyen. Denne implementasjonen gjør transaksjonen mellom informasjonssidene hos en kunde, leverandør, eller administatorsiden smidig.

4.2.2 Progressive disclosure

Målet ved brukeropplevelsen er å utvikle et system med god oversikt, og et godt organisert oppsett. For å sikre et godt organisert oppsett har gruppen valgt å ta i bruk patternet progressive disclousure for å kontrollere at oppsettet ikke overvelder brukeren med informasjon, men kun ser den viktigste informasjonen før bruker eventuelt velger å se detaljer[3]. Figuren, 4.4, demonstrerer hvordan gruppen har implementert progressive disclousure ved mailfunksjonaliteten. Mail vises med tidspunkt, emne og om noen har svart før bruker eventuelt velger å trykke på den for å vise resterende informasjon som mail tekst, svar og leserbekreftelser. Patternet er også brukt ved dashbordet, hvor oppsettet kun viser den viktigste informasjonen om ulike kundene før bruker kan trykke seg inn og se resterende informasjon. Øverste mail er åpnet, nederste mail er lukket.

[3] `http://ui-patterns.com/patterns/ProgressiveDisclosure`

Figur 4.4: Eksempel på Progressive disclosure

4.3 Universell utforming

Hensikten med universell utforming er å gjøre tjenester tilgjengelig for alle. Kommunal- og moderniseringsdepartementet publiserte i 2013 en forskrift som omhandler universell utforming av IKT-løsninger [7].

> *Forskriftens formål er å sikre universell utforming av informasjons- og kommunikasjonsteknologiske løsninger, uten at det medfører en uforholdsmessig byrde for virksomheten. Med universell utforming menes at utforming eller tilrettelegging av hovedløsningen i informasjons- og kommunikasjonsteknologi er slik at virksomhetens alminnelige funksjon kan benyttes av flest mulig*, jf forskrift om universell utforming §1 [7].

Universell utforming ved nye IKT-løsninger er lovfestet, og gjennom WCAG 2.0-standarden tilgjengeligjort gjennom tilsynet for universell utforming, har gruppen gjennom utviklingsprosessen tatt hensyn til disse kravene[6]. Minimumskravet for oppfyllelse av et universelt oppfylt brukergrensesnitt og design er i følge uutilsynets nettsider oppfyllelse av 35 av de 61 retningslinjene standarden opplyser om [6]. For utviklingsprosessen har gruppen valgt å oppfylle så mange av disse kravene det lar seg gjennomføre. Under listes et utvalg av de mest relevante hovedpunktene gruppen har tatt hensyn til gjennom prosjektet.

- Presentere innhold i en meningsfull og logisk rekefølge.
- Kontrastforhold mellom tekst og bakgrunn skal minimum være 4,5:1.
- Bruk av nyttige og tydelige sidetitler.
- Alle linker mål og funksjon fremstår tydlig i lenketekst.
- Endringer av fokus ved komponenter eller verdier ved skjemafelt medfører ikke automatisk betydlige endringer ved siden.

Dette er som nevnt bare et utvalg av de mest relevante hovedpunktene vi har valgt

å implementere. Under følger to mer detaljerte eksempler ved noen av kravene.

4.3.1 Klikkeflate/Navigasjon

Det er viktig at klikkbare objekter som knapper er lette å treffe både for personer med redusert presisjonsevne og for personer som bruker berøringsskjerm[4].

I gruppens implementasjon har dette blitt tatt hensyn til ved å at ingen knapper er for små. Knapper, samt tekst og andre elementer, skal responsivt tilpasses brukeren sin skjermstørrelse. Figur 4.5, demonstrerer knapper for å navigere til en kunde. Her er hele raden gjort klikkbar for å sørge for et større klikkområde for bruker.

Figur 4.5: Fokus ved knapper

4.3.2 Mobile løsninger

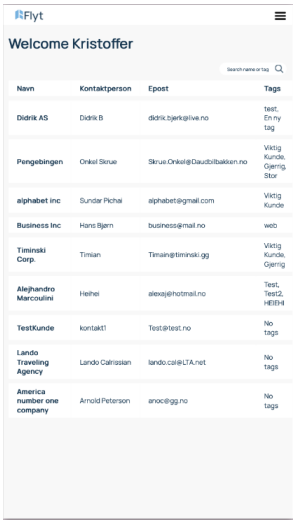
Kravspesifikasjonen definerer et behov for å utvikle systemet responsivt slik at det er tilgjengelig og tilpasset mobil og nettbrett, eller en minimert fane. WCAG 2.0-standarden definerer samme kriterium[5].

Gruppen har etter krav fra både oppdragsgiver og WCAG 2.0-standarden valgt å utvikle systemet med hensyn til et responsivt design gjennom hele utviklingsprosessen. Dette innebærer at alle funksjonaliteter og innhold automatisk tilpasses bruker sin skjermstørrelse. Figur, 4.6, viser hvordan designet av løsningen utformes ved PC og telefon.

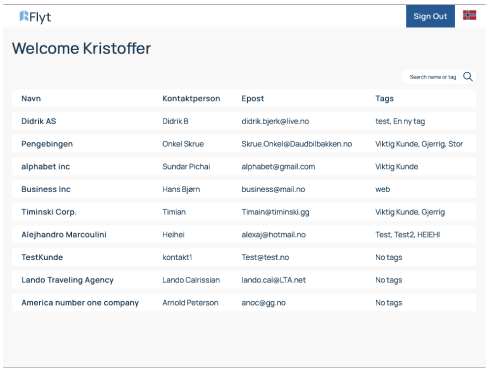
4.4 Designprinsipper

Gruppen har gjennom utviklingsprosessen av systemet forholdt seg til overordnede prinsipper for hvordan designet skal implementeres. Prinsippene har preget sluttresultatet og bidratt med målet om å oppnå et godt og effektivt brukergrensesnitt.

[4] `https://www.uutilsynet.no/regelverk/klikkeflate-navigasjon/211`
[5] `https://www.uutilsynet.no/regelverk/mobile-losninger/216`



(a) Mobilformat



(b) PC-format

Figur 4.6: Eksempel på responsivt utseende

4.4.1 Konsistens og repetisjon

Konsistens og repetisjon omhandler ett utseende og oppsett som gjennom hele systemet holdes likt, så lenge det er mulig[6]. Brukeren skal ha mulighet til å løse like problemer på like måter. Gruppen har valgt å implementere oppsettet til kunde- og leverandørsiden likt ettersom informasjon ved rollene inneholder likheter. Brukeren skal også kunne gjenkjenne elementer for deres funksjonalitet. Dette innebærer blant annet at knapper og søkefelt fremstår som det funksjonaliteten tilsier.

4.4.2 60-30-10 regelen

60-30-10 regelen omhandler fargesetting og et ønske om å holde web-applikasjonen stilren og enkel[7]. Regelen tilsier at 60% av siden dekkes av en primærfarge. Dette vil foreksempel si at bakgrunnsfargen vil dekke 60% av siden. 30% skal dekkes ved en sekundærfarge. I gruppens tilfelle vil dette innebære at komponenter dekkes ved denne fargen. 10% skal ha aksentfarge. Denne aksentfargen skal skille seg ut fra de andre fargene og blant annet brukes ved knapper eller andre elementer som skal fange brukerens oppmerksomhet. Fargebruken demonstreres blant annet ved figur 4.3. Se seksjon 4.5.2 for fargevalg.

4.5 Bestemmelser

Gjennom prosjektprosessen har gruppen stått ovenfor et utvalg av større bestemmelser for systemet. Valg av navn, farger, og logo er store beslutninger gruppen

[6] `https://uxdesign.cc/design-principle-consistency-6b0cf7e7339f`

[7] `https://uxdesign.cc/how-the-60-30-10-rule-saved-the-day-934e1ee3fdd8`

har brukt lang tid på å diskutere.

4.5.1 Navn

Oppdragsgiver delegerte gruppen ansvaret om å finne et passende navn til systemet. Valg av navn var en prosess gruppen valgte å utføre i flere ledd. Først samlet gruppen navneforslag over en lengre periode. Gruppen satte et krav om at navneforslagene skulle kunne assosieres med det systemet gruppen ønsket å utvikle. Etter å ha samlet navneforslag holdt gruppen en avstemning innad i gruppen hvor målet var å plukke ut topp tre navneforslag, og disse ble deretter presentert for oppdragsgiver. Etter diskusjoner med oppdragsgiver ble det bestemt at systemet skulle navngis *Flyt*. Flyt kan assosieres med enkelhet, lite friksjon, svevende og flytsone. Assosiasjonene passer de problemområdene applikasjonen løser, effektivisering og forenkelhet, og er derfor etter gruppens og oppdragsgiver ett godt passende navn til denne type system.

4.5.2 Farger

Farger for design av systemet er valgt etter diskusjoner med gruppen. Etter 60-30-10 prinsippet valgte gruppen ut tre hovedfarger for systemet. Valget av hovedfarge ble valgt etter inspirasjon fra TietoEvry sin nettside og logo, og ble derfor blå med hvit bakgrunn. Gruppen valgte ut en lys gråfarge for 60% delen, fargekode #f8f8f8. Denne delen skal i hovedsak dekke bakgrunnsfarge. 30% delen falt på en mørkere blåfarge, fargekode #063752, for dekorering av komponenter som blant annet navigasjonsbar og meny. I tillegg har gruppen plukket ut en lysere blåfarge for å hovedsaklig dekke de siste 10%, med fargekoden #62b3e5, og vil bli implementert i knapper og liknende. I tillegg vil gruppen dekorere knapper og fokusområder med relevante farger.

Figur 4.7: Farger

4.5.3 Logo

Utarbeiding av logo har hovedsaklig skjedd gjennom en forslagsprosess. Gruppen ønsket en logo med som kan assosieres med systemet, og samtidig bringer en dypere betydning. Det ble utarbeidet mange logoforslag, slik figur 4.8 demonstrerer.

Den ferdigstilte logoen, til venstre for ”Flyt” ved figur 4.9, skal hovedsaklig forestille en ”F” for Flyt. Samtidig skal de tre elementene ved designet forestille organisasjonen, kunden og leverandøren. Venstresiden fungerer som et bindeledd der

Figur 4.8: Logoutkast

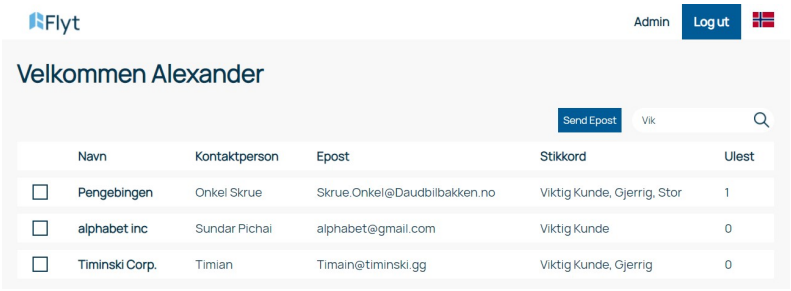
organisasjonen kobler de to mindre elementene til høyre, kunden og leverandøren, sammen og assosierer organisering og struktur. Fargebruken er inspirert av TietoEvry sin logo.

Figur 4.9: Logo

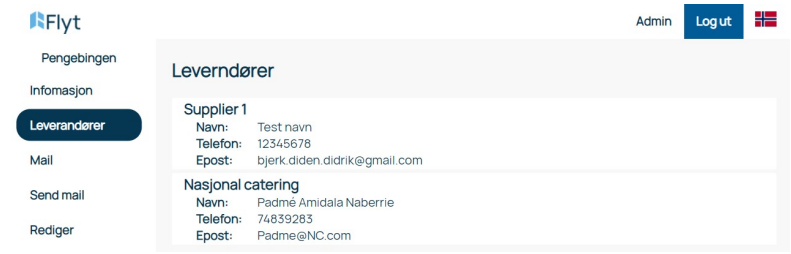
Logoen er utarbeidet etter en lengre prosess, og har blitt presentert og godkjent av oppdragsgiver med gode tilbakemeldinger.

4.6 Ferdigstilt brukergrensesnitt

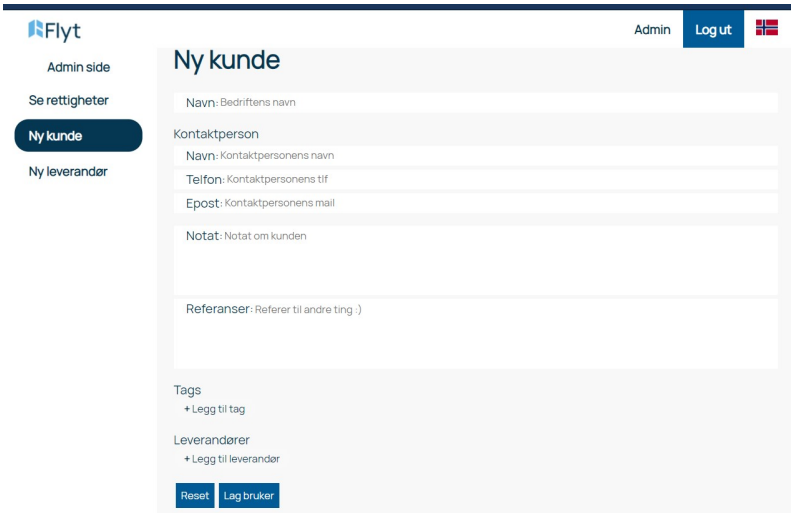
Nedenfor følger en visualisering av det ferdigstilte brukergrensesnittet ved webapplikasjonen.



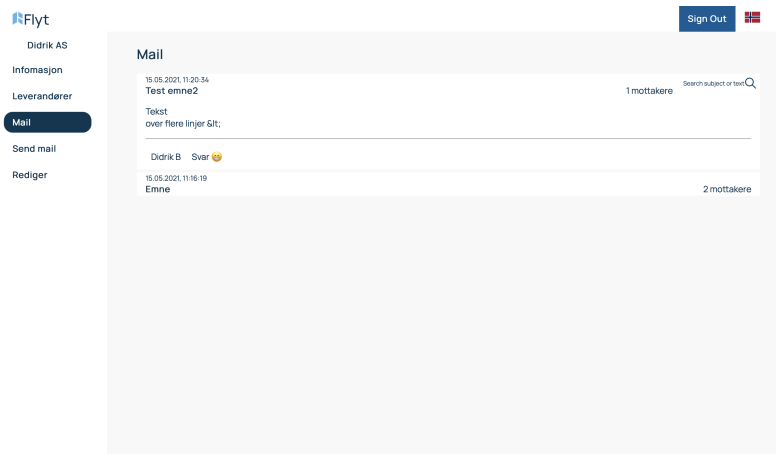
Figur 4.10: Dashbord



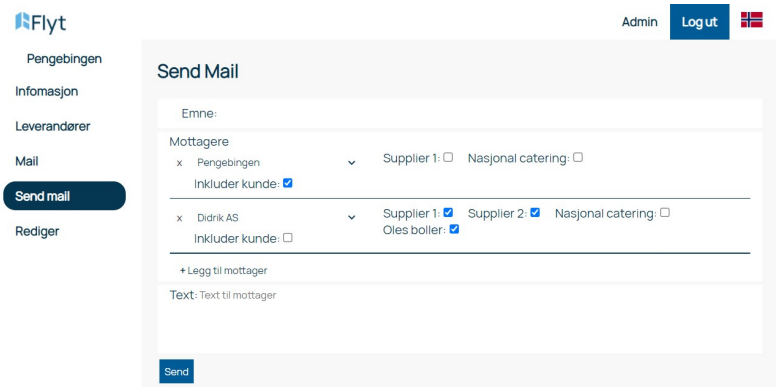
Figur 4.11: Kundeside



Figur 4.12: NyKunde-side i admin



Figur 4.13: SemMail-side



Figur 4.14: SendMail-side

Kapittel 5

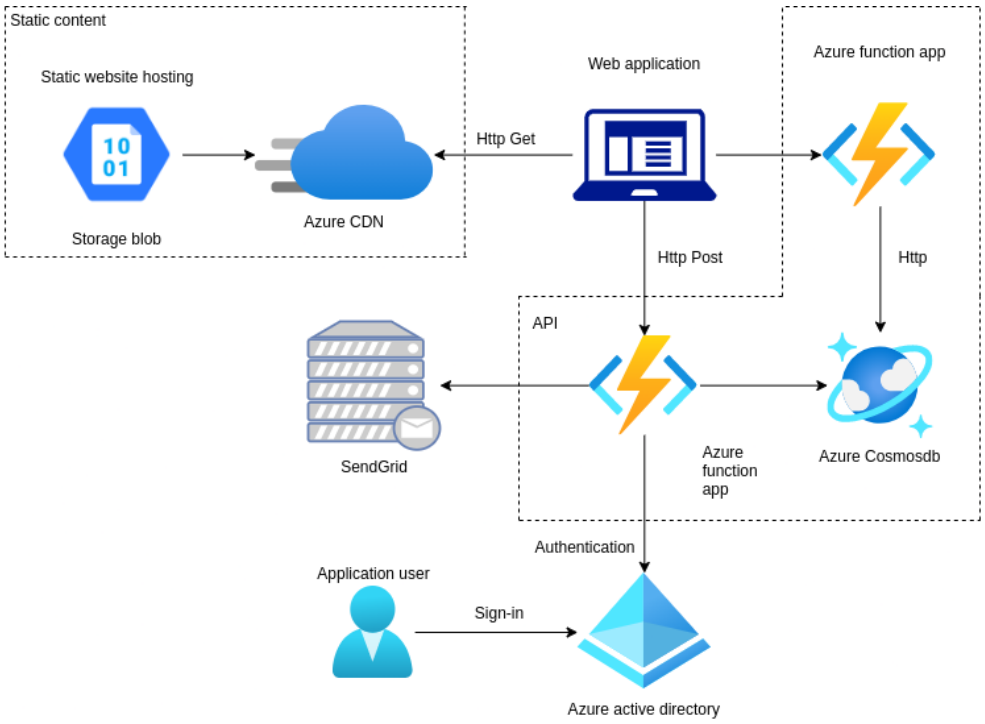
Teknisk design

Fra utarbeidet kravspesifikasjon til ferdig sluttresultat, har gruppen utarbeidet en plan for de tekniske aspektene ved systemet og hvordan disse skal løses i henhold til definerte krav og spesifikasjoner. Dette kapittelet omhandler hvordan utviklingsprosessen av det teknisk utformede designet av systemet er strukturert, og hvorfor gruppen har formet det overordnede systemet og systemkomponenter etter beskrevet struktur og metodikk.

5.1 Systemarkitektur

Den overordnede arkitekturen av systemet er utarbeidet etter tilhørende skytjenester av Microsoft Azure, se 6.1.1. Implementasjon av disse tjenestene ble definert som et krav fra oppdragsgiver i 1.8.2. Kravet ble definert tidlig, og gruppen brukte derfor oppstartsfasen på å kartlegge hvilke muligheter og tjenester Microsoft Azure tilbyr.

Etter diskusjoner innad i gruppen, og med oppdragsgiver, ble det bestemt at systemet skulle utvikles med en serverless arkitektur[8]. Ved å basere systemet på en serverless løsning har gruppen utformet den overordnede arkitekturen slik demonstrert med figur 5.1. Et begivenshetsdrevet system som gjennom HTTP-forespørsler utfører databasefunksjonalitet, autentiseringsfunksjonalitet, og tilbyr mailfunksjonalitet gjennom SendGrid. Figur 5.1 demonstrerer også hvordan Azure Blob Storage fungerer til oppbevaring av den statiske webapplikasjonen. Azure Content Delivery Network (CDN) lagrer regelmessig unna siste oppdaterte versjon av webapplikasjon i Blob Storage, og drifter webapplikasjonen under domenet Flyt.cloud. Modellen viser to Azure Functions applikasjoner, hvor den ene hovedsaklig er ansvarlig for API-et til systemet, og den andre administrerer mailfunksjonalitet for svar og leserbekreftelser. Alle tjenester og teknologier er ramset opp i kapittel 6.1.



Figur 5.1: Systemarkitektur (laget i draw.io)

5.2 Serverless

Som beskrevet i 5.1 har gruppen utarbeidet en serverless-arkitektur ved utformingen av systemet. Gjennom kravspesifikasjonen har oppdragsgiver definert diverse operasjonelle krav vedrørende brukerkapasitet og svartid, se kapittel 2.4. Samtidig har gruppen et ønske om å gjennomføre prosjektet moderne og kostnadseffektivt, ettersom reduksjon av kostnader ved utviklingen av systemet er definert som effektmål i 1.6.1. Serverless-strukturen gir gruppen mulighet til å tilpasse systemet etter disse kravene.

Det er flere grunner til at serverless arkitektur ble valgt for systemet. I dette kapittelet blir henholdsvis skalerbarhet og svartid diskutert, ettersom dette beskriver hvordan de tekniske aspektene til løsningen av de operasjonelle kravene er utformet. Videre drøfting av valget om serverless arkitektur blir drøftet i kapittel 9.2.1.

5.2.1 Skalerbarhet

Ved å benytte en serverless arkitektur vil systemet være skalerbart, både for opp- og nedskalering av den brukerkapasitet som ble spesifisert i de operasjonelle kravene, se 2.4. Tilgjengelig serverkapasitet er automatisk administrert av Microsoft

Azure, og en tilpasset forbruksbasert plan[1]. Denne planen innebærer i hovedsak at serverkapsitet og kostnader skaleres etter bruk. På denne måten vil systemet være skalerbart i henhold til brukerkapasitet, og kostnadseffektivt i henhold til Microsoft Azure sitt ”pay as you go”-prinsipp, som gjør at man kun betaler for de ressursene som har blitt brukt.

Forbruket kalkuleres ved å multiplisere gjennomsnittlig minneforbruk med tiden for gjennomføring av en funksjon i millisekunder[2]. Ved å bruke en forbruksbasert plan kan Microsoft Azure overvåke nettsidetrafikken og deretter tilpasse forbruk etter nødvendig kapasitet fra pågående trafikk. Dersom belastningen på prosessoren tilknyttet applikasjonen når et makspunkt, vil Microsoft Azure starte en identisk instans, og administrere trafikken mellom de eksisterende instansene.

Skalerbarheten til systemet er, som nevnt, kostnadseffektivt ved at man kun betaler for de ressursene som brukes[3]. Et potensielt problem med å bruke en forbruksbasert plan er at kostnadene potensielt kan drives opp av ondsinnede aktører[4]. Denne potensielle trusselen er løst ved å beskytte domenet til API-et bak Azure AD. Løsningen funger slik at en HTTP-forespørsel til en Azure Functions ikke blir utført dersom aktøren ikke er registrert i applikasjonens tenant og ikke kan gjengi gyldig access token. Dersom aktøren ikke er autentisert og kan gjengi gyldig access token, vil funksjonen returnere en feilmelding.

5.2.2 Svartid

For å oppfylle det operasjonelle kravet vedrørende svartid, skal svartid aldri overstige tre sekunder og gjennomsnittstiden skal ikke overstige ett sekund, se operasjonelle krav i 2.4. Ved serverless arkitektur kan det oppstå problemer dersom det ikke er aktivitet hos API-et.[5]. Dette kan medføre lengre svartid fra serveren, og i verste fall overstige grensen som ble satt i de operasjonelle kravene.

Oppstartsproblemet er et kjent problem med serverless arkitektur, og går under navnet, *cold start*[6]. Cold start skjer fordi vertsmiljøet deallokerer ressursene tildelt systemet etter en gitt periode uten aktivitet. For ny oppstart av systemet må vertsmiljøet først allokere applikasjonen til en server med ledig kapasitet. Deretter må serverens kjøretid starte før funksjonens kode kan eksekveres. For å løse problemet har gruppen implementert en funksjon som kalles med bestemte tidsintervaller, se kapittel 6.2.2. Dette medfører at systemet beholder de tildelte ressursene og ikke lenger er utsatt for denne trege oppstarten etter lengre bruk.

[1] https://azure.microsoft.com/nb-no/pricing/purchase-options/pay-as-you-go/

[2] https://azure.microsoft.com/en-in/pricing/details/functions/

[3] https://www.cloudflare.com/en-gb/learning/serverless/why-use-serverless/

[4] https://docs.microsoft.com/en-us/azure/azure-functions/security-concepts

[5] https://azure.microsoft.com/nb-no/blog/understanding-serverless-cold-start/

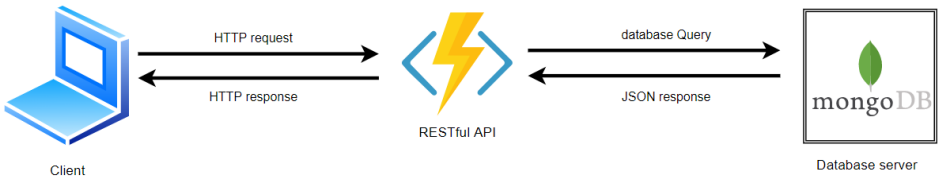
[6] https://azure.microsoft.com/nb-no/blog/understanding-serverless-cold-start/#:~:text=What%20is%20cold%20start%3F,must%20wait%20for%20their%20function

5.2.3 Azure Functions

Azure Functions er en FaaS tjeneste distribuert av Microsoft Azure for utvikling av blant annet begivenhetsdrevende funksjoner. Som nevnt i systemarkitektur, 5.1, benytter systemet seg av to Azure Functions applikasjoner, med ulike hensikter. Den ene Azure Functions applikasjonen er hovedsaklig ansvarlig for systemets API. Gjennom denne kjøres blant annet API-funksjoner gjennom utløste HTTP-forespørsler. Vertsmiljøet til systemet vil tilkalle funksjonen, som deretter utfører funksjonen og eksekverer koden til aktuelt funksjonskall. Ved å benytte Azure Functions vil serverlogikk være administrert av tjenesten, som gir gruppen mulighet til å fokusere på utviklingen av selve funksjonaliteten.

Azure Functions applikasjonen inkluderer flere funksjonaliteter, og alle eksekveres tilstandsløst[7]. Funksjonene må kunne utføres uten kjennskap fra tidligere hendelser eller historikk, og uten kjennskap til samtidige eksekveringer til andre funksjoner i samme system[8]. Alle funksjonene i en instans lever i samme minnområde, og derfor kan man allikevel opprette variabler som kan benyttes av flere funksjoner, selv om de ikke er satt opp til kommunisere med hverandre. Systemet benytter dette til å lagre databasetilkoblinger i en felles variabel. Dersom det forekommer en ledig databasetilkobling, sparer dette tid. Om dette ikke forekommer, dannes det en ny databasetilkobling. Siden funksjonene er lite avhengige av hverandre, regnes de for å ha lav kobling[9]. På grunn av dette er det enkelt å endre, fjerne og legge til ny funksjonalitet i API-et.

Gruppen bestemte seg også for å implementere API-et som et RESTful API, ettersom dette er moderne og skalerbart, og API-et er stateless[10]. Dermed har gruppen prøvd å opprettholde retningslinjene til et RESTful API. Figur 5.2 er en enkel illustrasjon av hvordan API-et kommuniserer med klienten og databasen.



Figur 5.2: REST API (laget i draw.io)

[7] https://docs.microsoft.com/en-us/archive/msdn-magazine/2019/august/azure-affairs-of-state-serverless-and-stateless-code-execution-with-azure-functions

[8] https://docs.microsoft.com/en-us/azure/azure-functions/functions-best-practices

[9] https://en.wikipedia.org/wiki/Loose_coupling

[10] https://restfulapi.net

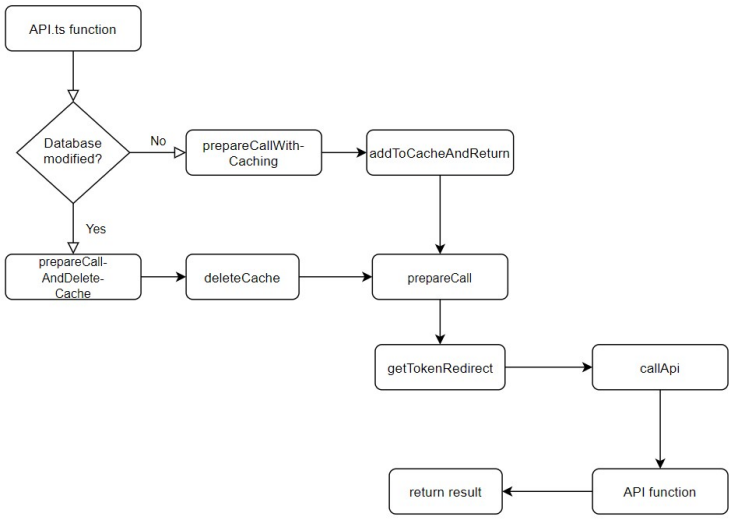
5.3 API

Utformingen av API-et er implementert etter serverless-strukturen beskrevet i de tidligere delkapitlene. For å forklare utarbeidingen av det fullstendige funksjonelle API-et har gruppen valgt å dele disse inn i en frontend- og en backend-del, hvor frontend-delen beskriver relasjonen mellom brukergrensesnittet og de implementerte backend-funksjonalitetene, i likhet med en controller i en MVC-arkitekturen. Backend beskriver utførelsen av HTTP-forespørsler gjennom backend-funksjonene.

5.3.1 Frontend

Frontend kaller API-et ved at det sendes en HTTP-forespørsel til en av backend-funksjonene. Avhengig om brukeren er autentisert og autorisert til å gjøre det det blir forespurt om, vil den motta en brukertilpasset respons. Sikkerheten vedrørende klarering blir bedre beskrevet i 6.2.1. Det implementerte API-et er enkelt å kalle direkte fra utarbeidede hentefunksjoner i webapplikasjonen.

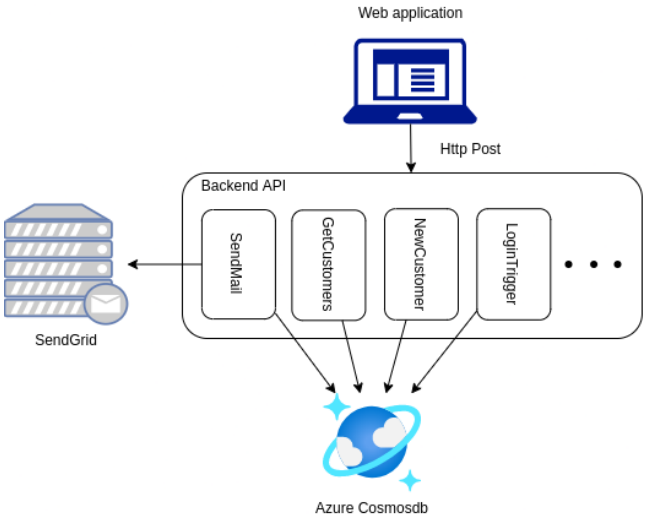
Figur 5.3 demonstrerer livsløpet til en API-funksjon fra den er kalt, til den er utført. Avhengig om det skal utføres modifiseringer i databasen blir en tilpasset *prepareCall*-funksjon kalt og oppdaterer eller sletter cache ved planlagt modifisering. Videre utfører *prepareCall* en forespørsel om en gyldig access token til autentisert bruker, denne prosessen beskrives videre i 6.2.1. For å utføre en HTTP-forespørsel må access token, endpoint-string for identifisering av backend-funksjonalitet og eventuell nødvendig tilhørende data medsendes *callApi*-funksjonen som parametere. Etter utførelse vil det returneres en JSON-formatert respons som resultat og en HTTP-statuskode.



Figur 5.3: API ved frontend-del (laget i draw.io)

5.3.2 Backend

Gruppen har utarbeidet backenden i henhold til bestemmelsene vedrørende implementasjon av en serverless-arkitektur. Hovedsaklig er API-et en rekke HTTP drevende funksjoner som utføres etter HTTP-forespørsel fra frontenden. Disse funksjonene er implementert som en Azure Functions App, beskrevet i 5.2.3. Funksjonene kan utføre henting og modifisering av databasen eller implementert mailfunksjonalitet gjennom *SendGrid*, slik vist i figur 5.4. I tillegg utfører funksjonene autentisering og autorisering ved hjelp av en medsendt *accessToken* før eventuell utførelse, videre beskrivelse i 6.2.2.

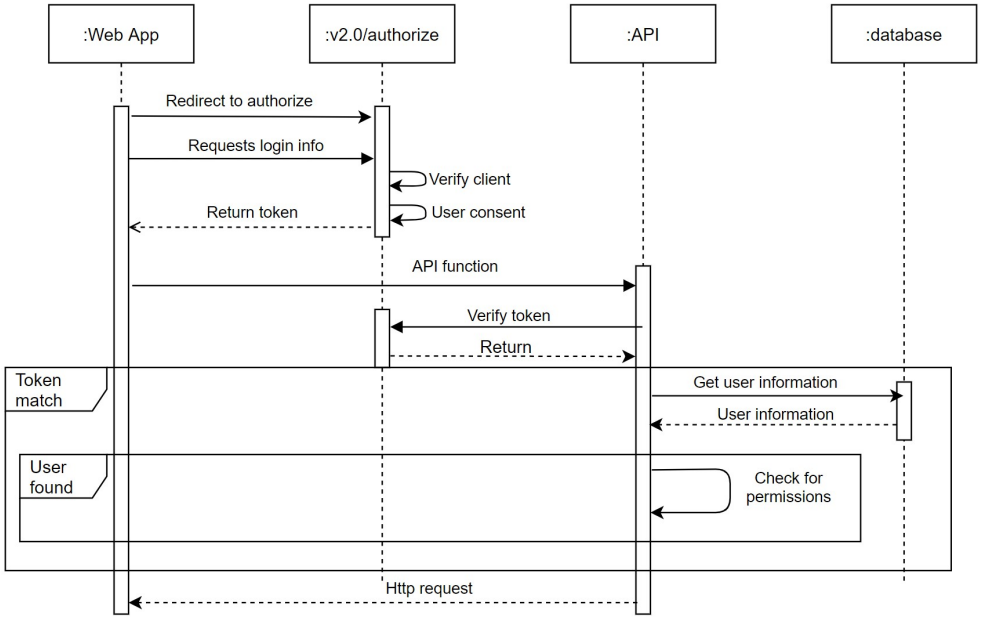


Figur 5.4: API ved backend (laget i draw.io)

Ved figur 5.4 er det overordnede bildet av kommunikasjonen mellom de ulike funksjonalitetene modellert. Figuren demonstrerer hvordan frontend kaller API-et gjennom HTTP-forespørselen. API-et utfører så forespørslene i databasen eller sender mail gjennom SendGrid.

5.4 Autentisering

Tilgang til systemet administreres av Azure Active Directory, Azure AD, og systemet er utarbeidet slik at kun ansatte i TietoEvry avdeling Brumunddal, og deres tilhørende kunder, skal ha tilgang til systemet. For å sikre at tilgangen begrenses til verifiserte brukere av systemet, implementeres autentiseringsmetodikk både før sendt HTTP-forespørsel, og ved eksekvering av HTTP-forespørsler som beskrevet i 5.3.2. Figur 5.5 demonstrerer autentiseringsprosessen til systemet gjennom et utarbeidet sekvensdiagram.



Figur 5.5: Autorisering sekvensdiagram (laget i draw.io)

Oppdragsgiver spesifiserte at verifiserte brukere skal administreres gjennom en tilpasset Azure Active Directory instans for systemet. Brukere sjekkes opp mot *0Auth* for verifisering av gyldig bruker, og får deretter returnert en access token hvis verifiseringen er gyldig. Ved et API kall blir access token sendt med og igjen sjekket for verifisering. Dersom access token er gyldig vil brukeren være autentisert. For å autorisere brukeren, hentes brukerens id fra fra access token etter at den har blitt dekryptert. Det sjekkes så hvilke rettigheter som er registrert på denne identifikasjonen i databasen. En mer detaljert forklaring vil bli demonstrert i neste kapittel, 6.

Ved å benytte Microsoft Azure og OAuth sin autentiseringmetodikk forsikrer gruppen seg om at autentiseringen i applikasjonen alltid vil være oppdatert i henhold til sikkerhetsprotokoller og nettleseroppdateringer. I tillegg vil autentiseringen utføres med en høy grad av sikkerhet, i henhold til Microsoft sine standarder[11]. Detaljert autentiseringsprosess blir beskrevet i kapitel 6.3.

Om derimot utviklerene skulle utvikle en egen autentisering, der brukernavn og passord blir lagret i en database, ville dette krevd mange ressurser. Teorien og teknologien rundt autentisering er i stadig utvikling og det ville vært krevende å sette seg inn i de forskjellige aspektene, blant annet hashing av passord, ha en trygg database og kryptert kommunikasjon. Ettersom autentisering er veldig viktig for applikasjonen, ville det krevd mange ressurser å ha et ordentlig autentiseringsystem, og ville måtte blitt endret på ved ett senere tidspunkt om autentiseringen

[11] `https://docs.microsoft.com/en-us/azure/app-service/overview-authentication-authorization`

ble utdatert. Derfor valgte gruppen å bruke ressursene som Microsoft tilbyr med Azure AD. Microsoft er et stort selskap med mange ressurser, og har kapasiteten til å opprettholde en god og trygg autentiseringfunksjonalitet.

5.4.1 Cookies

Det er også implementert en *silent login* funksjonalitet i systemet[12]. Silent login er en prosess hvor brukeren får tilgang til en token dersom brukeren har blitt autorisert i løpet av den siste timen. Denne tokenen blir byttet ut med en oppdatert token slik at brukeren får fornyet tiden før brukeren må autentiseres igjen[13]. Tokenen blir lagret som en cookie i nettleseren.

Ettersom det har blitt vanlig å blokkere *third-party cookies* for enkelte nettlesere vil ikke silent login fungere med alle nettlesere[14].

5.5 Database

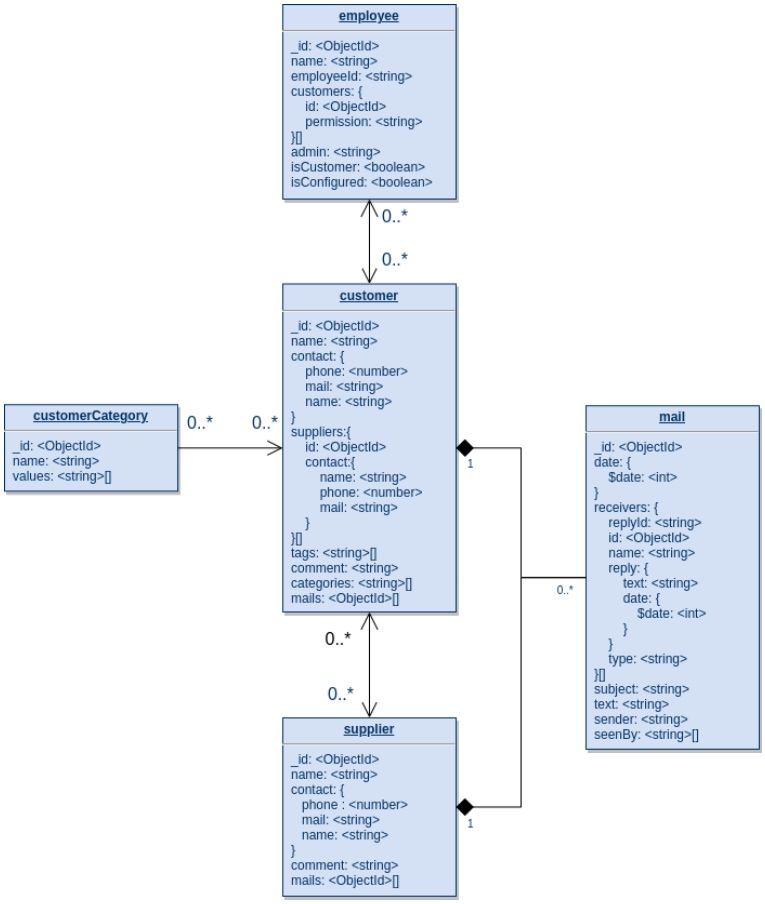
Ved utarbeidingen av database har gruppen benyttet NoSQL databasen Azure CosmosDB. Databasen bruker databaseprogrammet Mongodb. Det er et dokumentorientert, NoSQL program som lagrer data i JSON-liknende dokumenter[15]. Fordi applikasjonens frontend og backend er kodet i TypeScript, og databasen bruker MongoDB, skjer all form for databehandling og datalagring med data på samme format. Dette har gjort det enkelt for gruppen å forholde seg til applikasjonens informasjonsflyt.

Det som ansees for å være de beste praksisene når man designer et skjema for en Mongodb database, skiller seg fra de beste praksisene når man designer en relasjonsdatabase som SQL. Når man skal designe et skjema for en Mongodb database, gjør man dette med utgangspunkt i hva som gir best ytelse, og hvordan applikasjonen man lager den for fungerer. Dette er ganske annerledes fra en relasjonsdatabase, der det er klart definerte steg for hvordan skjemaet skal normaliseres. Vi har derfor kunnet designe en oversiktlig database med høy ytelse. Det at designet på databasen er tilpasset designet på applikasjonens frontend, bidrar med å gjøre det enklere for utviklere å sette seg inn i systemets helhet.

For å oppnå best mulig ytelse i en mongodb databasen, er det anbefalt at alle 1:1 relasjoner skal legges i samme dokument[16]. Dette gjenspeiles i databasens design, ved at alle relasjoner enten er 0:* eller 1:*, se figur 5.6. Å lagre data med 1:1 relasjoner i samme dokument reduserer antall *join* operasjoner som må gjennomføres av databasen, noe som reduserer belastningen på databasen. Dette

[12] `https://auth0.com/docs/sessions/cookies/spa-authenticate-with-cookies`
[13] `https://auth0.com/docs/libraries/auth0js`
[14] `https://auth0.com/docs/authorization/renew-tokens-when-using-safari`
[15] `https://www.mongodb.com/what-is-mongodb`
[16] `https://www.mongodb.com/basics/best-practices`

er bra for både ytelse og kostnad, fordi kostnadnsplanene i Azure Cosmosdb er basert på hvor bra ytelse man trenger, i tillegg til lagringsplass[17].



Figur 5.6: Database design (laget i draw.io)

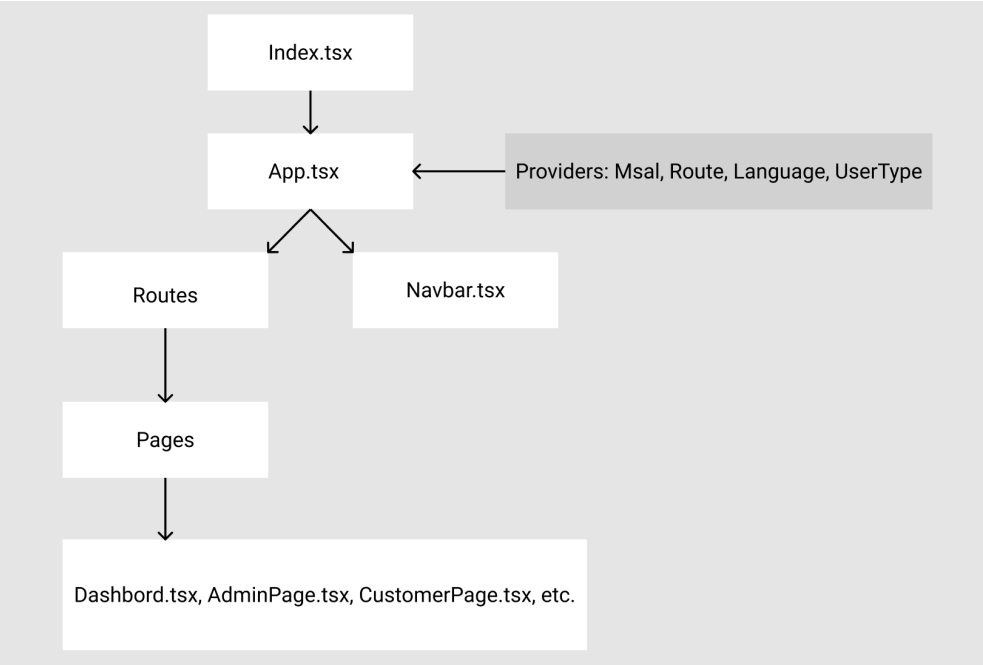
Som nevnt tidligere har gruppen hatt fokus på skalerbarhet, se 1.8.2, og vektlegger dette også når det gjelder databasen. Azure CosmosDB kan konfigureres med at stort antall forskjellige planer for skalering, ytelse og lagringsplass. De har statiske planer designet for både små og store applikasjoner, og dynamiske planer med automatisk skalering for applikasjoner som raskt må kunne respondere på økninger i gjennomstrømning[18].

5.6 React komponenthierarki

Ved utviklingen av webapplikasjonen har gruppen benyttet React for blant annet administrering av komponenthierarki og diverse globale variabler for utfor-

[17] `https://docs.microsoft.com/en-us/azure/cosmos-db/how-pricing-works`
[18] `https://docs.microsoft.com/en-us/azure/cosmos-db/provision-throughput-autoscale`

ming av brukergrensesnitt for systemet. Globale variabler blir administrert gjennom *useContext*-funksjonalitet, og gjennom *providere* plassert ved en komponent kan disse variablene benyttes ved alle underordnede komponenter av den komponenten den er integrert ved. Bruk av *useContext*-funksjonalitet blir beskrevet i kapittel 6.6.3.



Figur 5.7: Komponentarkitektur (laget i draw.io)

Figur 5.7 demonstrerer hvordan det overordnede komponenthierarkiet er strukturert. Ved App.tsx vil de ulike providerene bli implementert for å være tilgjengelig ved ønsket anledning i det lavere komponenthierarkiet.

- MSAL lagrer informasjon om bruker er logget inn eller ikke.
- Route holder orden på navigering i applikasjonen.
- Language gjør det mulig å skifte språk i alle komponenter.
- UserType holder orden på innlogget brukertype for et brukertilpasset grensesnitt.

Kapittel 6

Implementasjon

Gruppen har stått fritt ved valg av verktøy, språk, og annen metodikk for å løse oppgaven etter de spesifikasjoner og krav som defineres ved kravspesifikasjonen, se 2. Det eneste kravet fra oppdragsgiver vedrørende tekniske rammer ved utførlese av prosjektet, er å benytte Microsoft Azure, dokumentert ved 1.8.2. Det har fra oppstartsfasen vært et fokus innad i gruppen om å benytte moderne teknologi, så lenge dette er hensiktsmessig. Dette implementasjonskapittelet beskriver de tekniske detaljene for løsningsresultatet ved å vise til verktøy, utførelse, eksempler og drøftinger ved de implementerte løsningene.

6.1 Systemimplementasjon

I kapitel 5, har den overordnede systemarkitekturen og de ulike komponentsammensettingene blitt visualisert gjennom modeller og det ble gitt overordnende forklaringer. Systemet er bygd opp av mange, individuelt utviklede, komponenter. Hovedsaklig fungerer API-et som bindeledd mellom webapplikasjonen og backendfunksjonaliteter. CI/CD pipelines og Azure CDN fungert som bindeledd mellom utvikling og publisert resultat.

For utførelse av oppgaven har gruppen gjennom utviklingsprosessen benyttet ulike språk, rammeverk og teknologier. De neste delkapitlene lister de ulike teknologiene gruppen har brukt gjennom utviklingsprosessen, og beskriver kort hvilken betydning disse har hatt for systemet.

6.1.1 Microsoft Azure tjenester

Gruppen har i gjennom utviklingsprosessen benyttet et utvalg tjenester fra Microsoft Azure. Nedenfor følger en liste over brukte tjenester. Detaljert bruk av disse tjenestene blir beskrevet ved senere anledninger i dette kapitlet.

- Azure Functions, for API utvikling.

- Azure CDN, for distribuering av siste publiserte versjon av webapplikasjonen.
- Azure DNS Zone, for å administrere domenet, flyt.cloud.
- Azure Cosmos DB, for database for oppbevaring og organisering av data.
- Azure Blobstorage, for oppbevaring av siste publiserte versjon av webapplikasjonen.
- Azure DevOps, for administrering av CI/CD pipelines ved oppdatering av blobstorage til siste publiserte versjon av webapplikasjonen ved GitHub dersom pipelines gjennomføres uten feil.
- Azure Active Directory, for administrering av brukertilgang ved systemet.

6.1.2 Språk

Gjennom prosjektet har gruppen også benyttet et utvalg programmeringsspråk der det er sett hensiktsmessig. Følgende programmeringsspråk har blitt benyttet i løpet av utviklingsprosessen.

- TypeScript, og noe JavaScript, for frontend- og backend-utvikling.
- HTML og CSS, for utarbeiding av brukergrensesnitt.
- MongoDB Query Language (MQL), for håndtering av forespørseler vedrørende database.
- YAML, for konfigurering av pipelines ved utgivelser.

6.1.3 Biblioteker

For utviklingsprosessen av systemet har gruppen benyttet et utvalg biblioteker for implementering av diverse funksjonalitet. Følgende biblioteker er et utvalg av de viktigste implementerte bibliotekene. Disse er hovedsaklig installert ved bruk av Node Package Manager (NPM).

- React, for utarbeiding av brukergrensesnitt.
- Microsoft authentication library for JavaScript (MSAL.js), @azure/msal-browser og @azure/msal-react, for autentisering av brukere gjennom Azure Active Directory[1].
- JWKS-RSA, for å hente nøkler fra et JWKS, JSON Web Key Set.
- JSONWEBTOKEN, for å verifisere og dekryptere Access token[2].
- @azure/functions, for bruk av typedefinisjoner med Azure Functions.
- react/router og react/router-dom, for navigering av webapplikasjonen.

6.1.4 API

Gruppen har også benyttet et utvalg API-er for implementering av eksterne tjenester ved systemet. Følgende API er brukt ved utviklingsprosessen.

[1] `https://github.com/AzureAD/microsoft-authentication-library-for-js`
[2] `https://www.npmjs.com/package/jsonwebtoken`

- MongoDB APII, for kommunikasjon med databasen.
- Auth0, Azure AD authentification, for autentisering av brukere.
- Sanitize HTML, for å sikre at det ikke sendes utrygg HTML eller JavaScript kode.
- Typedoc, for generering av dokumentasjon av koden.
- SendGrid, for implementert mailfunksjonalitet.

6.1.5 Annen teknologi

Som nevnt tidligere, ble Typescript tatt i bruk, og gruppen brukte VSCode som kode editor. For å kjøre Typescript filene kjøres en kommando, som er installert via npm i tsc. Denne kommandoen kompilerer Typescript filen om til en Javascript fil som blir kompilert av Nodejs.

I tillegg har gruppen benyttet testingrammeverket Jest for utarbeiding av unit- og integrasjonstester gjennom prosjektet. Før Jest ble tatt i bruk, brukte gruppen Postman for å verifisere at API funksjonene fungerte som ønsket.

Backend-utviklerene lastet ned en Azure Functions-addon[3] i VSCode. Denne addonen gjør det enklere for utviklerne å lage, endre og iverksette nye funksjoner til API-et.

Gruppen har også utarbeidet API etter dokumentasjon for RESTful API[4]. REST er en arkitekturstruktur for distribuerte hypermedia systemer, og gruppen har oppfylt diverse krav for å kunne kalle API-et RESTful.

6.2 API

6.2.1 Frontend

API-et kobler webapplikasjonen mot backend-funksjonalitet, og den overordnede utarbeidingen og fremgangsmetodikken ved API-et er beskrevet ved 5.3.1. Gjennom denne implementasjonsdemonstrasjonen vil gruppen demonstrere utførelsen av den implementerte *GetCustomer*-funksjonaliteten.

Ved funksjonskall til API vil *prepareCallWithCaching* eller *prepareCallAndDeleteCache* kalles først, avhengig om funksjonen modifiserer databasen eller ikke. *GetCustomer*-funksjonen henter informasjon om en kunde, og vil dermed benytte *prepareCallWithCaching* ettersom den ikke utfører modifiseringer av databasen. *PrepareCallWithCaching* sjekker om funksjonkallet allerede er utført og informasjonen ligger i cache. Dersom dette er tilfellet vil den returnere det lagrede objektet.

Kodelisting 6.1 demonstrerer hvordan en typisk API-funksjon sender med en endpoint-string og eventuell tilleggsdata, og returnerer *prepareCallWithCaching*-funksjonen.

[3] `https://github.com/Microsoft/vscode-azurefunctions`
[4] `https://restfulapi.net`

```
export function getCustomer(id: string) {
  let customerId = {
    id: id,
  };
  return prepareCallWithCaching('GetCustomerData', customerId);
}
--------------------------------------------------------------------------------

async function prepareCallWithCaching(apiName: string, data = null) {
  let key = data?.id;
  if (!key) {
    key = apiName;
  }
  let cachedObject = await getFromCache(key);

  if (cachedObject !== null) {
    return cachedObject;
  }
  return addToCacheAndReturn(key, prepareCall(apiName, data));
}
```

Kodeliste 6.1: GetCustomer

Ved tom cache vil *prepareCallWithCaching*-funksjonen returnere *addToCacheAndReturn* og utføre *prepareCall*, for innhenting av gyldig access token og klargjøring av parametere før den senere HTTP-forespørselen. For å sende en forespørsel til backend må den aktuelle brukeren være autentisert for å få gyldig access token. Kodelisting 6.2 demonstrerer hvordan en parameterforespørsel sendes fra *callApi*-funksjonen, og gjennom et egenkonfigurert *msalInstance*-objekt. I objektet sjekkes det om forespørselen har tilgang til innhenting av access token.

```
export function getTokenRedirect(request): Promise<any> {
  //Henter innlogget konto ved brukernavn. Bruker må være innlogget for å kunne
  // Henter kontoinformasjon.
  const name = sessionStorage.getItem('UserName');
  request.account = msalInstance.getAccountByUsername(name);

  // Prøver å hente silentToken for bruker. Dersom det oppstår error vil dette
  // håndteres videre.
  return msalInstance.acquireTokenSilent(request).catch((error) => {
    console.warn('Silent token acquisition fails. Acquiring token using redirect.')
        ;
    if (error instanceof InteractionRequiredAuthError) {
      // Callback til interaksjon når silentToken feiler.
      return msalInstance
        .acquireTokenRedirect(request)
        .then((response) => { return response; })
        .catch((error) => { console.error(error); });
    } else { console.warn(error) }
  });
}
```

Kodeliste 6.2: Access Token

Dersom en gyldig access token har blitt innhentet, kan parameterene sendes med HTTP-forespørselen. Gjennom *CallApi*-funksjonen, demonstrert i kodeliste 6.3, sendes gyldig access token, endpoint string, og eventuell tilleggsdata med som parametere.

```
function callApi(endpoint, token, data) {
  const headers = new Headers();
  const bearer = 'Bearer ${token}';
  data = data ? JSON.stringify(data) : {};
  headers.append('Authorization', bearer);

  let options = {
    method: 'POST',
    headers: headers,
    body: data,
  };

  let status = null;
  return fetch(endpoint, options)
    .then((response) => {
      status = response.status;
      return response.json();
    })
    .then((response) => {
      if (response) {
        response['status'] = status;
        return response;
      }
    })
    .catch((error) => { console.error(error) })
}
```

Kodeliste 6.3: CallApi

Ved eksekvert *CallApi*-funksjonalitet vil resultatet returneres som et Json-objekt med HTTP-statuskode 200 for utførelse uten feil, eller aktuell HTTP-statuskode for feil ved utførelse.

6.2.2 Backend

Backenden er utarbeidet gjennom en Azure Functions app, slik beskrevet i 5.3.2. Backend består av 23 funksjoner, hvor 20 av disse utføres gjennom HTTP-forespørsler. Backenden er delt opp i såpass mange spesialiserte funksjoner, fordi det gir mening å dele opp Azure Function applikasjoner i så små og raske funksjoner som mulig. Det er fordi ressursbruken til funksjonene kalkuleres ved å gange tiden det tar å gjennomføre en funksjon med minneforbruket 6.2.2. Siden gruppen benytter HTTP-forespørsler for eksekvering av funksjoner er API-et utarbeidet etter de retningslinjene REST-API spesifiserer, og dermed kan også systemet karakteriseres som RESTful[5], beskrevet i 5.2.3.

[5] `https://restfulapi.net`

Med *GetCustomer*-funksjonen returneres aktuell kunde etter medsendt kundeidentifikasjon dersom den aktuelle brukeren har tilgang til kunden. Gjennom dekoding av medsendt access token fra frontend, ved hjelp av *jsonwebtoken*-biblioteket, kan funksjonen innhente kundeinformasjon. Funksjonen sammenlikner bruker-id fra den dekodede access token med kunden i databasen. Dersom bruker-id har tilgang på kunden blir informasjonen om kunden returnert.

Gjennom kontinuerlig bruk av access token for verifisering før eksekvering av HTTP-forespørsler, vil systemet sikre sensitiv informasjon bak riktig autorisering. Dersom kunden benytter en ugyldig access token, og dermed ikke er autentisert til funksjonaliteten, vil det returneres en feilkode. Funksjonen vil også returnere en feilkode hvis access token er gyldig, men brukeren ikke har tilgang til kunden identifikasjonsnummer. Alle funksjoner har selv ansvar for verifisering av access token før eventuell eksekvering av funksjonalitet.

Som beskrevet i kapittel 6.2.1 blir HTTP-forespørselen sendt gjennom *CallApi*-funksjonen fra frontend til backend. Kodeliste 6.4 beskriver utførelsesmetodikk gjennom pseudokode for en forenklet forklaring av prosessen.

```
export = (context: Context, req: HttpRequest): any => {
  /* All data kjøres gjennom en funksjon for html og javascript desinfisering */
  req.body = prepInput(context, req.body);
  if (req.body === null) {
    /* Avslutter med feilkode 400 */
  }
  /* Funksjon som klargjør access-token for autentisering/autorisering */
  let token = prepToken(context, req.headers.authorization);
  if (token === null) {
      /* Funksjonen avslutter med feilkode 400 hvis ikke access token sendes med */
  }

  const inputValidation = () => {
    /* Her ligger tester som sjekker at dataen er på riktig format */
    if (validInput) {
      /* Authorize() kalles som en callback funksjon,
         og sendes med en database tilkobling med leserettigheter. */
         connectRead(context, authorize);
    } else {
      /* Returnerer feilkode 400 hvis data er på feil format */
    }
  };

  const authorize = (db: Db) => {
   /* Her autentiseres brukeren*/
  };

  /* Brukes for filtrering av data */
  const query = {};

  /* Eksempel på å lage ny kunde */
  const functionQuery = (db: Db) => {
    db.collection(collections.customer).insertOne(query,
```

```
34        (error: any, docs: JSON | JSON[]) => {
35          /* Returnerer statuskode 200 hvis alt gikk som det skulle.
36          All dataen som returneres desinfisering for html og javascript kode*/
37          returnResult(context, docs);
38          context.done();
39        }
40      );
41    };
42    /* Første funksjon som kalles */
43    inputValidation();
44  };
```

Kodeliste 6.4: API-funksjon. Pseudokode

Kodeliste 6.4 demonstrerer et mønster for oppbygging av funksjonene. Funksjonen blir kalt og brukeren autoriseres, se kodeliste 6.6, og deretter eksekveres hovedmålet med funksjonen. Ved *GetCustomer*-funksjonen vil det utføres en MQL-spørring for innhenting av informasjon fra databasen.

Det blir også benyttet *sanitize-HTML* i begynnelsen av hver funksjon i *prepInput*-funksjonen. Sanitize-HTML er et API for å gå igjennom medsendt data, og lar kun data med trygge HTML tags bli eksekvert[6]. Sanitize-HTML kan også bli konfigurert til å ”desinfisere” CSS, men gruppen har ikke følt behov for denne implementasjonen. Eksempler på tags som kan være uønsket og farlige er ”<script>” og ”<style>”, mens godkjente tags kan for eksempel være ”
”, ”<h1>” og ”<table>”. Hensikten med implementasjonen av santize-HTML er å beskytte mot utrygge HTML tags og Cross-site scripting (XSS)[7]. XSS innebærer at ondsinnede brukere kan opprette script for manipulering, og få uønsket kontroll over deler av systemet. Sanitize-HTML, og sikker tilgang til data, skal sikre systemet for eventuelle informasjonslekkasjer.

For implementering av funksjoner har gruppen benyttet et Azure Functions tillegg i VSCode. Denne tilleggspakken har gjort det mulig å modifisere funksjonaliteten i API-et enkelt gjennom VSCode.

Som standard aksepterer Azure Functions Applications både http og https forespørler. Gjennom Azures portal har vi konfigurert applikasjonen så den krever at alle forespørsler gjøres over https, med siste tilgjengelige utgave av TLS[8]. Dette sikrer at alle data som sendes mellom frontend og backend er kryptert.

6.2.3 Svartid

Som beskrevet i 5.2.2 kan det oppstå problemer som treg oppstarttid til funksjonene. Dette forårsakes av implementert serverless struktur basert på Azure Functions, og er et vanlig problem med serverless arkitektur. Dette løses gjennom en

[6] https://www.npmjs.com/package/sanitize-html
[7] https://en.wikipedia.org/wiki/Cross-site_scripting
[8] https://docs.microsoft.com/en-us/azure/azure-functions/security-concepts

implementert funksjon som utløses med et satt tidsintervall på tre minutter, ettersom tilkoblingen til databasen er satt til å brytes etter tre minutter og 20 sekunder fra siste eksekvering[9].

```
export default (context: Context, myTimer: any) => {
  checkDbConnection(context, clientRead);
  // Kobler til database for å hindre treg oppstart.
  const timeStamp = new Date().toISOString();

  if (myTimer.isPastDue) {
    context.log('Timer function is running late!');
  }
  muligens fjerne V
  context.log('Timer trigger function ran!', timeStamp);

  connectRead(context, () => context.done(), true);
};
```

Kodeliste 6.5: Tidsutløst oppstartfunksjon

Kodelisting 6.5 demonstrerer hvordan funksjonen er implementert. Ved løsning av dette problemet, lager denne funksjonen en tilkobling til databasen. Fordi alle funksjonene i Azure Functions applikasjonen eksekveres i et felles minneområdet kan denne tilkoblingen til databasen benyttes av de andre funksjonene i systemet. Dermed vil svartiden reduseres, og det sikrer en kontinuerlig rask svartid ved funksjonskall slik de operasjonelle kravene definerer, se kapittel 2.4.

6.3 Autentisering

Autentiseringen til systemet består av to stadier, en i frontend og en i backend. Frontend-autentiseringen løses ved å bruke Microsoft authentication library (MSAL)[10] som dirigerer bruker til en innloggingside hvor verifiserte brukere i Azure AD kan logge inn. Ved innlogging blir brukeren validert og brukeren blir navigert til tilpasset startside, dette er demonstrert i kodeliste 6.8.

Det er først når brukeren kaller en funksjon fra API-et, via en HTTP-forespørsel, at autentiseringen i backend blir aktivert. Brukerens access token blir sendt med i HTTP-forespørselen og blir verifisert. Om den er godkjent, så kjøres det en spørring til databasen som finner ut om brukeren har lov til å utføre resten av funksjonen. Hvis det i løpet av autentiseringen viser seg at brukeren ikke har lov til å utføre funksjonen, blir funksjonen stoppet og det returneres en feilmelding samt statuskoden 401[11].

Kodeliste 6.6 demonstrerer autentiseringsprosessen av en bruker med skriverettigheter i databasen. Den gir også ett glimt av autoriseringen, som vil bli forklart

[9] https://docs.microsoft.com/en-us/azure/azure-functions/functions-scale
[10] https://github.com/AzureAD/microsoft-authentication-library-for-js
[11] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/401

videre i 6.3.1.

```
const authorize = (db: Db) => {
   /* Access token verifiseres og dekodes */
   verify(token, getKey, options, (err: any, decoded: Decoded) => {
     if (err) {
     /* Avbrytes med status 401 og 'token not valid' */
       errorUnauthorized(context, 'Token not valid');
       return context.done();
     } else {
       db.collection('employee') // Sjekker om bruker har admin rettigheter
         .find({ employeeId: decoded.preferred_username })
         .project({ admin: 1 })
         .toArray((error: any, docs: { admin: string }[]) => {
           if (error) {
             /* Avbrytes med status 401 og 'token not valid' */
             errorQuery(context);
             return context.done();
           } else {
             if (docs[0].admin === 'write') {
               //databasetilkoblig med skriverettigheter
               connectWrite(context, functionQuery);
             } else {
               errorUnauthorized(context, 'User dont have admin permission');
               return context.done();
             }
           }
         });
     }
   });
 };
```

Kodeliste 6.6: API autentisering eksempel

Autentiseringsprosessen benytter både access tokens og ID-tokens. Etter en bestemt tidsperiode, på en time, vil access tokenen bli ugyldig, og brukeren må da autentisere seg selv igjen. ID-tokens blir strengt tatt ikke tatt i bruk i løpet av autentiseringen, men blir istedenfor bruk etterpå. ID-tokene gjør det enklere å identifisere brukere etter at de har blitt autentisert. For øyeblikket brukes ID-tokens bare for å gi applikasjonen informasjon om hvilken rolle brukeren har. Denne rollen blir bare brukt første gang brukeren logger seg på nettsiden, og utifra hvilken rolle brukeren har, blir brukeren enten satt til å være en kunde eller ikke. Det er mulig at brukerens ID-token kan ha flere verdier, men det er hovedsakelig bare en verdi som forteller om brukeren er en kunde eller ikke.

For implementasjon av autentisering i systemet har gruppen benyttet 0Auth 2.0 med *hybrid flow*. Hybrid flow, også kalt authorization code flow, er vanlig å bruke i webapplikasjoner[12]. Valget ved å bruke OAuth var bilateralt. Først og fremst, så er det gjort rede for å bruke OAuth til å autentisere brukere for Azure AD. I tillegg

[12] `https://docs.microsoft.com/en-gb/azure/active-directory/develop/v2-oauth2-auth-code-flow`

så tilbyr OAuth en god balanse av sikkerhet og brukervennlighet, som gjør at dette passer godt for gruppen.

```
if (employee == undefined) {
              decodedToken.roles.forEach((role) => {
                if (role == 'customer') {
                  isCustomer = true;
                  result['isCustomer'] = true;
                }
              });
```

Kodeliste 6.7: API autorisering eksempel

6.3.1 Autorisering

Når brukeren logger inn i webapplikasjonen vil brukeren kalle en *callLogin* funksjon for å bekrefte at vedkommende er autentisert. Dersom brukeren er autentisert og ingen feilmeldinger oppstår, vil brukeren få returnert et Json-objekt med autoriseringdata til applikasjonen. Gruppen har implementert følgende metodikk for delegering av riktig brukertilgang, se kodeliste 6.8.

```
useEffect(() => {
    async function fetchAccountInfo() {
      setLoading(true);                    //Setter loading til true.
      setIsError(false);                   //Setter error til false.
      try {
        let info = await callLogin();      //Henter autoriseringsdata.
        switch (true) {
            .
            .   //Dersom bruker er konfigurert, har leserettigheter ved admin, og
            .   // er første pålogging vil brukertilgangen bli "AdminReadFirst"
          case info.isConfigured && info.admin === 'read' && info.firstLogin:
            userTypeChange('AdminReadFirst');
            break;
            .
            .
            .
          default:  //Dersom ikke stemmer vil den bli satt til ikke konfigurert.
            userTypeChange('NotConfigured');
        }
      } catch (error) { //Hvis error settes error til true.
        setIsError(true);
      }
      setLoading(false); //Loading settes til false.

    }
    fetchAccountInfo(); //Kjører funksjonen ovenfor.
  }, []);              //Kjøres ved første rendering av startsiden.

  return (
    <div>
      {isError ? (
        <div> Det er feil </div>
```

```
33        ) : Load ? (
34          <Loading />
35          )
36          .
37          .
38          .
39         : userType === 'AdminReadFirst' ? (  //Blir delegert adminrettigheter.
40          <Admin />
41          .
42          .
43          .
44        ) : (
45          <PageNotFound /> // Hvis ikke stemmer blir pageNotFound vist.
46        )}
47      </div>
48    );
```

Kodeliste 6.8: Autorisering av brukertilgang

Kodelistingen 6.8 demonstrerer delegering av brukerrettigheter til brukerrollen administrator med leserettigheter og første innlogging. Det blir først utført en switch hvor brukerrollen defineres ut ifra medsendt data. Deretter blir brukeren navigert til riktig startside. Brukerrollen blir lagret i den globale variablen userType, og blir benyttet ved andre komponenter for å sikre riktig tilgang og visning.

I backend, så skjer autorisering veldig raskt og enkelt. Etter at brukeren er autentisert, sjekkes det for hvilke rettigheter brukeren har, og om rettighetene er tilstrekkelig for det funksjonen ønsker å gjøre, utføres resten av funksjonen. Se 6.6, linje 18 til 24.

6.4 CI/CD

Gjennom CI/CD og IaC, infrastructure as code, sikrer gruppen kontinuerlig utgivelse av fungerende kode. Beskrevet metode for utgivelse er dokumentert ved 7.1.

CI er den første pipelinen som kalles ved en oppdatering av GitHub-repositoriets hovedgrenen. Gjennom en spesifisert YAML-fil går pipelinen gjennom installasjoner, testing og bygging av prosjektet. Dersom det ikke oppstår feil blir det publiseres en ny artifact-zipfil med applikasjonen. Som medfører at nettsiden er oppdatert.

```
1 trigger:
2 - main
3
4 pool:
5   vmImage: 'ubuntu-latest'
6
7 steps:
8 - task: NodeTool@0
```

```
  inputs:
    versionSpec: '10.x'
  displayName: 'Install Node.js'

- script:
    npm install
  displayName: 'npm install'
  workingDirectory: 'Frontend'

- script:
    set "REACT_APP_STAGE=development" && npm test
  displayName: 'npm test'
  workingDirectory: 'Backend'

- script:
    set "REACT_APP_STAGE=development" && npm test
  displayName: 'npm test'
  workingDirectory: 'Frontend'

- script:
    set "REACT_APP_STAGE=development" && npm run build
  displayName: 'npm build'
  workingDirectory: 'Frontend'

- task: CopyFiles@2
  displayName: 'Copy files'
  inputs:
    sourceFolder: 'Frontend/build'
    Contents: '**/*'
    TargetFolder: '$(Build.ArtifactStagingDirectory)'
    cleanTargetFolder: true

- task: ArchiveFiles@2
  displayName: 'Archive files'
  inputs:
    rootFolderOrFile: '$(Build.ArtifactStagingDirectory)'
    includeRootFolder: false
    archiveType: zip
    archiveFile: $(Build.ArtifactStagingDirectory)/$(Build.BuildId).zip
    replaceExistingArchive: true

- task: PublishBuildArtifacts@1
  displayName: 'Publish Build Artifacts'
  inputs:
    pathtoPublish: $(Build.ArtifactStagingDirectory)
```

Kodeliste 6.9: CI pipeline

YAML-filen, kodeliste 6.9 inneholder kommandoene kloning av repositorium, installering av npm biblioteker, gjennomføring av tester, en build-kommando, og en publiseringskommando av konstruert *artifact* i denne rekkefølgen.

CD, tilkalles ved trigger etter ny publisert artifact fra CI-pipeline. Denne pipelinen har som hovedoppgave å oppdatere blobstorage med det nye publiserte innholdet

fra artifact-zipfilen. På samme måte som CI-pipelinen er denne konfigurert til å sletter det gamle innholdet og publiserer en dekomprimert utgave av det nyeste publiserte artifacten til blobstorage.

6.5 Database

Ved implementasjon database har gruppen tatt hensyn til diverse sikkerhetkrav. Gjennom tilkoblingen til databasen vil autorisering benyttes for å sikre riktig informasjon til riktig bruker, og data blir sikkert oppbevart med et geo-redundant oppsett.

6.5.1 Tilkobling til databasen

Azure Cosmos DB har noen sikkerhetskrav for de som skal få tilgang til databasen. Kravene er hovedsakelig at alle som skal ha tilgang til databasen må være autentisert og kommunikasjonen skal gå gjennom en sikker kobling kryptert med SSL. Brukere blir autentisert i API-ets funksjoner og blir tildelt en databasetilkobling som etableres ved å bruke en privat connection string[13]. For å koble databasen til applikasjonens backend, har vi fulgt Microsofts råd om å legge til databasens *connection strings* som application settings[14] i backenden[15]. Dette gjør at de ikke er synlige i applikasjonens klidekode, og bare tilgjengelig for brukere som er lagt til i Azure med rettigheter til å endre applikasjonen. Tilkoblingen som etableres til databasen ved å bruke connections strings, er kryptert med SSL. Det er totalt to connection strings, en for leserettigheter og en for skriverettigheter, hvor disse administrerer muligheten for modifisering av databasen. Backenden er satt opp slik at kall til databasen aldri har flere rettigheter enn de trenger.

For å koble seg til databasen, kaller API-funksjonene en av to funksjoner. For funksjoner som bare skal lese data, kalles connectRead, men for funksjoner som også skal endre på data, kalles connectWrite. Disse funksjonene bruker forskjellige connection strings, som ble nevnt i et tidligere avsnitt. Hensikten med to ulike funksjoner er å ikke gi API-funksjoner tilgang til mer enn det som er nødvendig for at den skal utføre sin funksjonalitet. Dette er gjort i henhold til prinsippet om minste privilegier[16]. For eksempel har getCustomer-funksjonen kun leserettigheter, fordi den kun henter informasjon fra databasen.

```
1 export function connectRead(context: Context, callback: (arg0: any) => void,
      overrideTest = false) {
2   // Om overrideTest er true, så lages det en ny kobling uansett
3   if (clientRead == null || overrideTest) {
```

[13] https://docs.mongodb.com/manual/reference/connection-string/

[14] https://docs.microsoft.com/en-us/azure/azure-functions/functions-app-settings

[15] https://docs.microsoft.com/en-us/azure/azure-functions/functions-add-output-binding-cosmos-db-vs-code?pivots=programming-language-javascript

[16] https://docs.microsoft.com/en-us/azure/lighthouse/concepts/recommended-security-practices

```
4     MongoClient.connect(uriRead, config, (error: any, _client: any) => {
5       if (error) {
6         context.log('Failed to connect read client');
7         context.res = {
8           'status': 500,
9           'body': 'Failed to connect read client',
10        };
11        return context.done();
12      }
13      clientRead = _client;
14
15      context.log('Connected read client');
16      callback(clientRead.db(DBName));
17    });
18  } else {
19    callback(clientRead.db(DBName));
20  }
21 }
```

Kodeliste 6.10: Dannelse av databasetilkobling

Ved kodelisting 6.10 blir det demonstrert hvordan en tilkobling med leserettigheter blir opprettet, hvis det ikke finnes en åpen tilkobling fra tidligere. Dersom det oppstår en feil vil funksjonen returnere en feilmelding.

6.5.2 Geo-redundant lagring

Konfigurert via Azure tas det også en sikkerhetskopi hver fjerde time. Alle sikkerhetskopiene blir lagret i åtte timer før de blir slettet. All dataen i sikkerhetskopiene er lagret i "storage blobs",[17] der all dataen er lagret som ustrukturert data[18]. For å øke sikkerheten til informasjonen som blir lagret i databasen, tas Geo-redundant storage (GRS)[19] i bruk. GRS går ut på at dataen blir lagret i gruppens primære datasenter, men også i et annet datasenter. Datasenterene er plassert langt unna hverandre, men fortsatt sentralt innenfor regionene sine. Datasenteret som blir brukt av applikasjonen ligger i Amsterdam, Nederland og "par-datasenteret" ligger i Dublin, Irland. Ved å oppbevare data i to ulike datasentre vil ikke uforutsette hendelser med et av datasentrene ha negativt utslag for lagret data. Dette vil si at ved strømproblemer eller naturkatastrofer ved et av datasenterene, vil det andre datasenteret være upåvirket av hendelsen.

6.6 Brukergrensesnitt

Ved implementasjon av brukergrensesnitt har gruppen benyttet JavaScript-biblioteket React[20]. React er et bibliotek utviklet for å konstruere brukergrensesnitt for we-

[17] https://docs.microsoft.com/en-us/azure/cosmos-db/configure-periodic-backup-restore
[18] https://docs.microsoft.com/en-us/azure/storage/blobs/storage-blobs-introduction
[19] https://docs.microsoft.com/en-us/azure/storage/common/storage-redundancy
[20] https://reactjs.org/

bapplikasjoner. Det sørger for et enkelt oppsett av komponentrelasjonering, testing, og inkluderer flere løsninger gruppen har valgt å benytte for utviklingen. De neste delkapitlene demonstrerer hovedelementer ved oppbygningen av brukergrensesnittet.

6.6.1 NPM

NPM er tatt i bruk for å administrere avhengigheter, og et stort register av *packages* og *node moduler*. En node module er en JavaScript-fil, og en package er en katalog med flere moduler. NPM er veldig populært, og inkluderer over 1.3 millioner packages[21].

Gruppen har brukt NPM ettersom den tilbyr en brukervennlig måte å installere andre utvikleres kode. Gruppens har brukt NPM for å innhente ferdig utarbeidet, og offentlig publisert funksjonalitet der det er hensiktsmessig. Etter at brukeren har installert node.js, kan utvikleren bruke kommandoen *install* for å laste ned moduler, se kodeliste 6.11.

```
1 npm install <package/module name>
```

Kodeliste 6.11: Installasjon av package/node modul via NPM

Etter installasjonen av ønsket NPM-materiale vil funksjonalitet plasseres i en lokal node-package katalog, og være tilgjengelig for bruk. Ved nytt oppsett av koden vil en *"npm install"*-kommando installere alt nødvendig NPM-materiale.

6.6.2 React-komponenter

Den implementerte løsningen av brukergrensesnittet er komponentorientert, i henhold til React sin komponent- og oppbygningsstruktur, se 5.6. Gruppen har implementert komponenter ved blant annet navbar, sidemeny, knapper, og hele sider.

Hver komponent inneholder HTML og CSS for å vise komponenten. Den kan også inneholde Typescript/Javascript kode som variabler og funksjoner, eller andre komponenter. React-komponenter varierer i størrelse fra enkle knapper, til menyer og hele sider. Ved endringer av data på en side vil React kun rerendere de komponentene som rammes av endringene. Dette gjør nettsiden responsiv og rask, fordi kun de nødvendige delene av siden blir oppdatert.

Et komponentorientert oppsett gjør det også enkelt å gjenbruke komponenter og funksjonaliteter flere steder i applikasjonen, og tilpasse bruken ved hjelp av parametere. Parametere blir sendt med som props, et parameterobjekt med tilhørende data. Kodelisting 6.12 demonstrerer et enkelt eksempel på hvordan props har blitt brukt i implementeringsprosessen.

[21] `https://blog.npmjs.org/post/615388323067854848/so-long-and-thanks-for-all-the-packages`

```
interface PersonData {
  name: string;
  age: number;
}

function DisplayPerson(props: PersonData) {
  return (
    <div>
      <p>Navn: {props.name}</p>
      <p>År: {props.age} </p>
    </div>
  );
}

//bruker komponenten
<DisplayPerson name={"Alexander"} age={22} />
```

Kodeliste 6.12: Eksempel på props

6.6.3 React Hook

React Hook er en funksjonalitetstjeneste som gjør det mulig å bruke tilstandsvariable uten å implementere klasser. Det gjør det også mulig å administrere livssyklusfunksjonalitet ved funksjonorienterte komponenter.

React useContext er en React Hook, og muliggjør administrering av tilstand i ulike komponenter. Gruppen har gjennom prosjektet benyttet dette for å administrere blant annet språk, og brukerrolle ved innlogget bruker. Denne tilstandsadministreringen gjør det mulig å modifisere variable gjennom en *Reducer*, og ta i bruk oppdaterte variable ved ønskede komponenter. Kodelisten 6.13 demonstrer hvordan webapplikasjonen oppretter en global tilstand, i form av useContext, hvordan det implementeres, og hvordan det blir tatt i bruk.

```
// Lage "language context" med spesifisert interface og standardspråk.
interface ITodosContextData {
  userLanguage: string;
  dictionary: any;
  userLanguageChange: (c: string) => void;
}

export const LanguageContext = createContext<ITodosContextData>({
  userLanguage: 'en',
  dictionary: dictionaryList.en,
  userLanguageChange: () => {},
});

------------------------------------------------------------------------

// Legge til context-funksjonalitet i komponenter.

// Importer LanguageContext.
```

```
19 import { LanguageContext } from '../../context/language/LangContext';
20
21 // Importer variable og funksjon ved bruk av useContext.
22 const { userLanguage, dictionary, userLanguageChange } = useContext(LanguageContext
       );
23
24 // Kan bruke variablene og funksjonen i komponenter.
25 <button onclick={() => ( userLanguage === 'en' ) ? setLanguageChange('no') :
       setLanguageChange('en')}> dictionary.name </button>
```

Kodeliste 6.13: Eksempel på bruk av React UseContext Hook

I kodeliste 6.13 opprettes det først et interface med tilhørende definisjoner av parameterne. Det opprettes en context-funksjonalitet gjennom *LanguageContext*, hvor variablene blir tildelt standardverdier. Ved opprettet context kan denne importeres og bli tatt i bruk ved diverse komponenter, som beskrevet ved nederste del av kodelisten.

I tillegg til tilstandsadministrering har gruppen benyttet andre React Hooks ved henting av data, og livssyklusfunksjonalitet. Gjennom *useEffect* kan det spesifiseres hvilke funksjoner som skal eksekveres ved første render, eller om de skal oppdateres ved endring av en spesifisert variabel. Kodelisting 6.14, demonstrerer bruk av *useEffect* ved to bruksområder. I kodelistingen blir også *useState* benyttet. *useState* fungerer som en React Hook for lokal tilstandsadministrering ved komponenter.

```
1  // UseState for lokal tilstandshåndtering ved komponent.
2    const [customers, setCustomers] = useState(null);
3    const [search, setSearch] = useState('');
4    const [filter, setFilter] = useState(null);
5
6  // Bruker useEffect til å hente data fra "getEmployee". UseEffect kjøres //  kun
       ved oppstart ettersom "[]" er tom.
7    useEffect(() => {
8      const fetchName = async () => {
9        let customers = await getEmployee();
10       setCustomers(customers);
11       setFilter(customers.customerInformation);
12     };
13     fetchName();
14   }, []);
15
16 // Bruker useEffect til å oppdattere listen etter søking i liste. Denne
17 //   kjøres hver gang search blir oppdatert ettersom dette defineres ved
18 //   "[search]".
19   useEffect(() => {
20     const filtered = (e) => {
21       const filteredList = customers.customerInformation.filter((customer) => {
22         const tag = customer.tags.toString().toLowerCase();
23         const name = customer.name.toString().toLowerCase();
24         const currsearch = tag + name;
25         return currsearch.indexOf(search.toLowerCase()) !== -1;
26       });
27       setFilter(filteredList);
```

```
28      };
29      if (customers !== null) {
30        filtered(search);
31      }
32    }, [search]);
```

Kodeliste 6.14: React UseEffect og UseState Hook

Kodelistingen 6.14 demonstrerer hvordan gruppen har benyttet livssyklusfunksjonaliteten ved useEffect til å administrere oppdateringer av komponenten. Ved kodelistingen kjøres første fetch av ”getEmployee” ved oppstart av komponenten. Søkefunksjonaliteten derimot blir bare kjørt, og renderer komponenten, dersom search-variabelen endrer seg. Grunnen til at useState-variabler brukes istedenfor vanlige variabler er at useState-variabler rerendrer komponentene som bruker den når den blir endret[22]. Mens vanlige variabler kun vil endre data uten å oppdatere komponenten.

Fordelene ved bruk av React Hooks er hovedsaklig oversiktlighet i et funksjonsorientert komponentoppsett. React Hooks fungerer som en erstatning for det tradisjonelle klasseoppsettet, og skal i tillegg være mer allsidig og fungere smidig over oppsettet av komponenter.

6.6.4 React Router

For navigering av webapplikasjonen har gruppen benyttet React Router for tilordning av URL.

```
1  // Demonstrasjon av route til Dashboard.
2  const Routes = () => {
3    return (
4      <Switch>
5        .
6        . // /dasboard viser dashboard- og customerpagekomponent
7         <Route path='/dashboard' component={Dashboard} />
8         <Route path='/customerpage' component={CustomerPage} />
9        .
10       .
11     </Switch>
12   );
13 };
```

Kodeliste 6.15: Delegering av URL-adresse

Kodelisting 6.15 viser hvordan dashbordkomponenten blir tilordnet en *Route path*, "/dashboard". Dersom brukeren går til "/dashboardurlen vil den komme til dashbordkomponenten. Ved en provider i App.tsx blir *routes* delegert til resten av webapplikasjonen ved de underliggende komponentene i hierarkiet.

[22] `https://reactjs.org/docs/hooks-state.html`

6.6.5 React Hook Form

React Hook Form, er et React Form bibliotek som simplifiserer implementasjon av skjema i brukergrensesnittet[23]. Et skjema er en samling av input-elementer, som tekst- og sjekkbokser[24]. Skjema er mye brukt ved implementeringen av brukergrensesnittet, og brukes blant annet ved sending av mail, og modifiseringer av kunder, leverandører og ansatte.

For å effektivisere implementeringen av brukergrensesnittet har React Hook Form gjort det mulig å raskt utarbeide funksjonelle løsninger. React Form inkluderer funksjonalitet for validering og oppdatering av tilhørende data ved modifisering av andre felt. Dette har i større grad simplifisert implementasjonprosessen. React-hook-forms ble valgt over andre løsninger fordi det er lightweight, og det unngår unødvendig rerendring av siden. Det er også lett å legge til validering og å oppdatere felt basert på andre felt.

Skjema virker ved at input-elementer blir registrert ved en useForm-funksjon. Denne funksjonen har oversikt over aktuelle verdier, og bestemmer når registrerte elementer skal rerendres. Når brukeren er ferdig med redigeringen og klikker på endre-knappen, blir verdiene i skjema sendt fra useForm til en API-funksjon for videre utførelse, for eksempel oppdatering av database.

Figur 6.1: Eksempel på skjema laget med react hook form

I kodelisting 6.16 demonstreres funksjonaliteten ved implementasjon av et statisk skjema for modifisering av en ansatt, ved bruk av React Hook Form. Skjemaet er statisk fordi man vet hvor mange og hvilke input-elementer skjemaet inneholder fra start. I eksempelet har en ansatt et navn, alder, og en array med navnene på kundene sine. Figur 6.1 demonstrerer hvordan eksemplet kan se ut på nettsiden.

```
1  // Verdiene som brukes på skjemaet
2  interface IFormValues {
3    employeeName?: string;
4    employeeAge?: number;
5    customers?: { name: string }[];
6  }
7
8  // React komponent med skjema for endreing av kunde
9  function EmployeeEditForm(props: IFormValues) {
10   // Setter default verdi om det er sendt med
```

[23] https://react-hook-form.com/

[24] https://www.w3schools.com/html/html_forms.asp

```
  const { control, register, handleSubmit } = useForm<IFormValues>({
    defaultValues: {
      employeeName: props?.employeeName,
      employeeAge: props?.employeeAge,
      customers: props?.customers,
    },
  });

  return (
    <form onSubmit={handleSubmit(updateEmployee)}>
      <p>Ansatt info:</p>
      <input {...register('employeeName')} placeholder='Navn' />
      <input {...register('employeeAge')} placeholder='Alder' />

      // For å redigere kundene til en ansatt
      <EditCustomer />

      <button type='submit'> Endre kunde </button>
    </form>
  );
}
```

Kodeliste 6.16: Statisk React hook form eksempel

Kodelisting 6.16 demonstrerer hvordan useFrom settes opp med standardverdier fra props. Input-elementene for navn og alder blir registret i skjema. I bunn av skjema ligger en utfør-knapp som sender informasjonen. Siden man ikke vet på forhånd hvor mange kunder en ansatt har, blir det brukt et dynamisk skjema for å generere kundenavn-feltene. Komponenten *<EditCustomer />* inneholder et dynamisk skjema, og koden er demonstrert i kodeliste 6.17.

```
// Lager dynamisk skjema
const { fields, append, remove } = useFieldArray({
  control,
  name: 'customers',
});

// React komponent som endrer kundenavn
function EditCustomer(){
  return(
    <p>Kunder</p>
    // Redigering av kundene til en ansatt
    {fields.map(({ id }, index) => {
      return (
        <div>
          <button onClick={() => remove(index)}>x</button>
          <input
            key={id} {...register(`customers.${index}.name` as const)}
            placeholder='Kunde navn'
          />
        </div>
      );
    })}
```

```
23
24      // Legger til kunde
25      <button type='button' onClick={() => append({})}>
26        Legg til kunde
27      </button>
28    }
```

Kodeliste 6.17: Dynamisk React hook form eksempel

Kodelisting 6.17 implementerte koden for modifisering av hvilke kunder en ansatt er ansvarlig for. *Fields*-variabelen blir brukt til å holde informasjon om kundenavnene. Dersom den ansatte ikke har kunder vil variabelen forbli tom og det eneste som vises er knappen for å legge til nye kunder.

Dersom den ansatte har kunder, eller brukeren trykker på legg til kundeknappen, blir modifiseringsfeltene vist. Det er også en knapp ved siden av input-elementene for å fjerne eksisterende kundenavn.

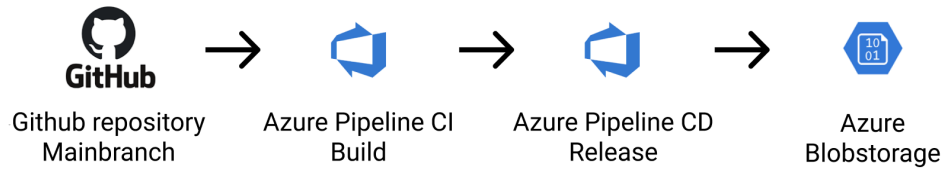
Kapittel 7

Utrulling

Kravspesifikasjonen definerer at systemet skal implementeres som en webapplikasjon, og webapplikasjonen vil være tilgjengelig gjennom *Flyt.Cloud*. Systemet distribueres gjennom skytjenesten Microsoft Azure, hvor vedlikehold ved bruk av deres tjenester er deres ansvar, men allikevel har gruppen hatt en målsetting om å legge enkelt til rette for videreutvikling og vedlikehold av kode ved systemt. Gruppen har utviklet systemet med fokus på å gjøre systemet skalerbart, og har benyttet en smidig og enkel distribueringprosess for å publiseringer oppdateringer.

7.1 Utgivelser

Gruppen har gjennom utviklingsprosessen benyttet Azure DevOps ved utgivelser. Azure DevOps har en innebygget pipelinefunksjonalitet med mulighet for å konstruere IaC-filer, infrastruktur som kode. IaC er en rask, effektiv, og trygg metode for å distribuere siste publiserte versjon av systemet[1]. Pipelinen gjennomfører nødvendige installasjoner, *builds*, testinger og andre eventuelle spesifikasjoner. Implementasjon og beskrivelse av metoden blir beskrevet i kapittel 6.4. Etter dette blir lagringsplassen for webapplikasjonen, Azure Blobstorage, oppdatert etter siste vellykkede utgivelse fra CD pipeline.



Figur 7.1: CI/CD (laget i draw.io)

[1] https://azure.microsoft.com/nb-no/services/devops/pipelines/

Prosessen fra en oppdatering av hovedgrenen i GitHub-repositoriet, til en oppdatering ved blobstorage-repositoriet er automatisert gjennom en utløser ved pipelinen, slik figur 7.1 demonstrerer og det blir beskrevet i kapittel 6.4. Triggeren ved GitHub-repositoriet starter en pipelineprosess hver gang aktuell hovedgren blir oppdatert.

7.1.1 Flyt.Cloud

Etter siste utgivelse til Azure Blobstorage er en ferdig bygget versjon av webapplikasjonen tilgjengelig gjennom en Azure Storage Container, hvor den distribueres som en statisk nettside[2]. Målet med utgivelsen er allikevel å oppdatere domenet Flyt.Cloud for siste publiserte versjon i Azure Blobstorage.

For å publisere systemet gjennom Flyt.Cloud, bruker systemet Azure tjenesten Azure CDN. Azure CDN er et innholdsleveringsnettverk som med jevne mellomrom sjekker Azure Storage Container for oppdateringer, og lagrer det siste oppdaterte innholdet på et distribuert nettverk av servere. Azure CDN brukes også for administrering og automatisk oppdatering av nettsidens SSL sertifikat, for å sikre at alt innhold leveres gjennom en sikker HTTPS tilkobling[3]. Administrering av domenet, Flyt.cloud, styres gjennom Microsoft Azure tjenesten, DNS Zone.

7.2 Tilgjengelighet

Systemet er tilgjengelig gjennom domenet Flyt.cloud, hvor det distribueres gjennom Microsoft Azure som beskrevet over, 7.1.1.

Tilgangen til systemet administreres gjennom Azure AD hvor tilgang er restriktert brukere registret i systemets tenant har tilgang til innlogging. Tenanten er en dedikert og isolert instanse av Azure AD. DennAzure ADen administreres av brukere med administratorrettigheter i Azure AD. Ettersom oppdragsgiver allerede benytter Microsoft sine tjenester og har eksisterende Microsoft-konti, vil admin kunne invitere eksisterende brukere eller opprette nye.

Applikasjonens oppetid avhenger først og fremst av skytjenestene den benytter. Tjenestene som brukes og påvirker oppetiden er CDN, Azure Functions, Azure Cosmos DB og Azure AD. Følgene garantier vedrørende opptid er dokumentert hos Microsofts Service-level Agreement (SLA) ved de øvrige skytjenestene.

- 99.90% for Azure CDN[4]
- 99.95% for Azure Functions[5]
- 99.99% for Comsos DB[6]

[2] `https://azure.microsoft.com/nb-no/blog/azure-storage-static-web-hosting-public-preview/`
[3] `https://docs.microsoft.com/en-us/azure/cdn/cdn-overview`
[4] `https://azure.microsoft.com/en-us/support/legal/sla/cdn/v1_0/`
[5] `https://azure.microsoft.com/en-us/support/legal/sla/functions/v1_1/`
[6] `https://azure.microsoft.com/en-us/support/legal/sla/cosmos-db/v1_4/`

- 99.99% for Azure AD[7]

Ved de operasjonelle kravene ble det spesifisert et krav om oppetid på minimum 99,9%, se 2.4. I følge den opplistede dokumentasjonen vil det være flere forbehold for å ta hensyn til. Ved et verst tenkelig scenario vil det operasjonelle kravet ikke bli oppfylt dersom tjenestenes nedetid er uavhengige av hverandre. Fordi flere av tjenestene kjøres i samme datasenter, er det derimot sannsynlig at de opplever nedetid samtidig.

$99.90\% * 99.95\% * 99.99\% * 99.99\% = 99.83\%$

Regnestykket ovenfor viser oppetid for hele systemet ved verst tenkelig scenario. Ettersom det er et krav om å oppnå 99,9% oppetid, vil ikke systemet oppfylle dette kravet. Dersom tjenestenes oppetid er avhengige av hverandre vil oppetiden være høyere, (da er) ikke gruppens estimering av oppetid være korrekt. Dersom det estimeres etter best tenkelig scenario vil tjenestene oppleve nedetiden i samme tidsrom. Best mulig scenario vil derfor være at oppetiden er 99,90%, i henhold til dokumentert oppetid for Azure CDN. Samtidig er best tenkelig scenario utarbeidet etter de verste scenarioene for hver enkel tjeneste, og oppetiden kan dermed også være høyere enn 99,90%. Gruppen har vurdert oppetid etter verste scenario dersom verste scenario ved alle tjenestene skulle inntreffe. Dette vil dermed være svært usannsynlig, men ettersom situasjonen kan inntreffe vil ikke systemet kunne garantere en oppetid på mer enn 99,83%, altså oppfylles ikke det operasjonelle kravet om oppetid på 99,9%.

For å tilfredstille det operasjonelle kravet ville gruppen være nødt til å la være å benytte disse tjenestene. Ettersom det ble dokumentert som et teknisk krav å benytte Microsoft Azure sine tjenester for å utarbeide systemet, se 1.8.2, vil det derimot ikke være mulig å oppfylle begge kravene uten å la være å benytte deler av de tjenestene Azure tilbyr.

7.2.1 README

Gruppen har opprettet en README-fil i GitHub-koderegisteret for instruksjoner om hvordan systemet kjører lokalt for eventuelle videreutviklere av systemet. Det er detaljert definert ned til hvilke spesifikk kommando, og inneholder generell informasjon om nødvendige justeringer i koden.

7.3 Skalerbarhet

Gjennom kravspesifikasjonen defineres systemet til å skulle administrere 100 samtidige brukere, se 2.4. Det skal også legges til rette for at videre skalerbarhet enkelt kan bli implementert med Microsoft Azure.

[7] `https://azure.microsoft.com/en-us/support/legal/sla/active-directory/v1_1/`

Systemet er utviklet ved hjelp av serverless teknologi og Azure Functions for å legge til rette for en dynamisk automatisert skalerbarhetsprosess hvor man bare betaler for bruken av ressursene. En mer detaljert beskrivelse av skalerbarhet er dokumentert i 5.2.1.

7.4 Vedlikehold

For skytjenestene applikasjon benytter, er både maskinvare, vertsmiljø og kjøretidsmiljø administrert av Microsoft. Det er derfor de som har ansvar for at de systemene holdes oppdatert og vedlikeholdt[8].

Applikasjonens kildekode inneholder et stort antall packages og node moduler6.6.1. Det hender at det oppdages sikkerhetshull i disse pakkene, og at de da må oppdateres for å sikres. Vertsmiljøet applikasjonen kjøres i vil ikke oppdatere dem automatisk, vedlikehold av disse må dermed gjøres manuelt.

Dersom det er ønskelig å videreutvikle systemet er koden godt dokumentert med markeringsspråket JSDoc. Dette gjør koden lettleselig for nye utviklere. I tillegg til dokumentasjon direkte i koden er det også generert Hypertext Markup Language (HTML)-filer med oversikt over funksjonene, hvordan de brukes, hvilke parametere de har og hva de returnerer.

[8] `https://azure.microsoft.com/en-us/updates/azure-functions-updates/`

Kapittel 8

Testing og kvalitetssikring

Gjennom prosjektutviklingsperioden har gruppen hatt som mål å utvikle et CRM-system i henhold til prosjektbeskrivelsen, som oppfyller prosjektmålene for oppgaven, dokumentert i kapittel 1.4 og 1.6. Gruppen har implementert testing for å forsikre at implementasjoner fungerer til sin hensikt. Det er implementert integrasjonstesting, unit testting, og gjennomført brukertesting. Gjennom kapittelet vil testingen vedrørende prosjektet demonstreres, drøftes, og konkluderes med effekten testingen hadde på resultatet.

8.1 Testing av backend

Gjennom testingen av backendfunksjonalitet har gruppen implementert integrasjonstesting. Denne type testing omhandler interaksjon mellom ulike elementer i systemet, og gruppen har utarbeidet integrasjonstester for blant annet interaksjon med databasen.

8.1.1 Integrasjonstesting

Testene for API-et har som mål å forsikre at koden som ble skrevet i funksjonene returnerer data som stemmer overens med forventet data. I tillegg blir funksjonene testet for om brukeren har tilgang til systemet og ressursen, og om data brukeren sender med til funksjonen stemmer overens med forventet data.

```
1  export function prepareContext() {
2    let context = {
3      res: { status: null, body: null },
4      log: (txt) => null,
5      done: null,
6    };
7
8    context = {
9      res: { status: null, body: null },
10     log: (txt) => null,
```

```
    done: () => (context.done = true),
  };
  return context;
}

export let httpRequest = {
  headers: { authorization: '' },
  body: {},
};
```

Kodeliste 8.1: Context og httpRequest ved testing

Mocking er essensielt for å utføre tester som har eksterne avhengigheter. At et objekt blir mocket vil si at man bytter ut innholdet i objektet, men de samme parameterene blir fortsatt mottatt av objektet. Dette gjør at man kan bli kvitt eksterne avhengigheter og man kan redefinere hva objektet eventuelt skal returnere. Mocking skjer bare under testing og har dermed ingen effekt på web-applikasjonen.

Alle testene følger samme struktur. Først blir objektene *context* og *httpRequest* mocket, se kodeliste 8.1. Deretter blir funksjonen man ønsker å teste kalt med context og httpRequest som parametre. *Timeout*-funksjonen stenger alle databasetilkoblinger når API-funksjonen fullføres, eller om databasetilkoblingene er åpne lengre enn ti sekunder. Deretter sjekkes context for data, om dataen stemmer overens med det som er forventet, er testen vellykket.

```
// Mocket verify-funksjon
export const verify = (token, b, c, callback) => {
  if (token?.length === 0) {
    callback('Error no id', {});
  } else if (token == 'test') {
    callback('Error no id',
        { name: 'randomName', roles: [], preferred_username: 'name' });
  } else {
    token = token.replace('Bearer ', '');
    callback(null, { name: 'randomName', roles: [], preferred_username: token });
  }
};
--------------------------------------------------------------------------------
// Original verify-funksjon
export function verify(
    token: string,
    secretOrPublicKey: Secret | GetPublicKeyOrSecret,
    options?: VerifyOptions,
    callback?: VerifyCallback,
): void;
```

Kodeliste 8.2: verify-funksjonen

Inne i alle funksjonene som startes av en HTTP forespørsel blir *verify*-funksjonen mocket. Verify-funksjonen vil originalt dekode tokenen om til informasjon om tokenens eier. Etter at funksjonen har blitt mocket returnerer den fortsatt informasjon om eieren, men det er alltid en bestemt bruker som blir returnert. Denne bru-

keren varierer utifra parametrene som blir sendt med og er tilpasset hvilken test det er. For eksempel om testen skal undersøke hva som skjer når ingenting skal gå galt, sendes det med mailen til en bruker som er i databasen og har riktige rettigheter. Se kodeliste 8.2 for sammenlikning av mocket og original verify-funksjon.

Kodeliste 8.3 er et eksempel på to tester av en funksjon. De fleste funksjonene har fra to til åtte tester hver, antallet avhenger av hvor mange forskjellige situasjoner som kan oppstå under eksekvering.

```
describe('User credentials', () => {
 test('Should work as expected', async (done) => {
   let context = prepareContext();
   let request = httpRequest;
   request.headers.authorization = 'oyvind.husveg@kundeboss.onmicrosoft.com';
   request.body['id'] = id;

   GetSupplierData(context as any, request as any);
   await timeout(context);

   expect(context.done).toEqual(true);
   expect(context.res.body).toHaveProperty('_id', id);
   expect(context.res.status).toBe(200);
   done();
 });

 test('Token is empty/null', async (done) => {
   let context = prepareContext();
   let request = httpRequest;
   request.headers.authorization = '';
   request.body['id'] = id;

   GetSupplierData(context as any, request as any);
   await timeout(context);

   expect(context.done).toEqual(true);
   expect(context.res.body).toBe('no token');
   expect(context.res.status).toBe(400);
   done();
 });
});
```

Kodeliste 8.3: Test example

8.1.2 Postman

Postman er et program som gjør det lett for utviklere å sende HTTP forespørseler til API-et. I starten av prosjektet ble Postman brukt for å manuelt teste API-funksjonene, men gruppen fant ut at det var bedre måter å gjøre det på. Mest merkverdig, integrasjonstester via Jest, som beskrevet ved 8.1.1.

8.2 Testing av frontend

Gruppen startet opp med å teste frontend-komponenter ved unit-testing. Testingen av komponentene tok lengre tid, og gruppen følte ikke testingen gav gode nok resultater i forhold til tiden som ble lagt investert. Gruppen utarbeidet allikevel et par unit-tester ved et utvalg komponenter.

8.2.1 Unit-testing

Ved utarbeiding av frontend-testing har gruppen testet for at komponenten inneholder de elementene den er ment å inneholde. Ved kodelisting 8.4 blir en testing av *about*-siden demonstrert.

```
1  describe('about page', () => {
2
3  // Other import and mock props
4    afterEach(cleanup)
5
6    test('Expected IDs', () => {
7      const { debug, getByTestId, getByText } = render(<About />);
8
9      getByTestId("Title");
10     getByTestId("Center1");
11     getByTestId("Center2");
12   });
13
14   test('English is working', () => {
15     const { debug, getByTestId, getByText } = render(<About />);
16
17     getByText("About");
18     getByText("TestTestTest", {exact : false});
19   });
20 });
```

Kodeliste 8.4: Unit-test example

Testingen kompilerer komponenten og sjekker at alle elementer er inkludert. Deretter sjekker den at engelsk, som er standardspråk ved webapplikasjonen, er fungerende.

Gruppen utarbeidet denne testingen av komponentene ved et par komponenter, men fant ut at tidsbruken dette tok ikke var verdt det i forhold til resultatet av testene.

8.3 Brukertesting

Ved gjennomføring av brukertesting har gruppen lagt vekt på testing av det grafiske og logiske aspektet ved systemet. Målsettingen for utviklingsprosessen av systemet er å utvikle et oversiktlig og organisert system, slik resultatmålet tilsier

se kapittel 1.6.2. Ved denne testprosessen ønsker gruppen å identifisere eventuelle forbedringer og/eller ulogiske implementasjoner ved systemet. Testene gjennomføres ved å delegere ulike scenario til deltakere, hvor de uten veiledning skal navigere og gjennomføre aktuelt scenario. Deltakerne ble bedt om å tenke høyt under utførelsen av scenarioet, og det ble stilt refleksjonsspørsmål vedrørende deres opplevelse og oppfatning av systemet. Testingen ble gjennomført ved et tilfeldig utvalg deltakere med ulik alder, kjønn og kjennskap til liknende CRM-system. Gruppen håper med dette å tilegne seg verdifull innsikt fra et ulikt utvalg deltakere.

Brukertestingen ble utført mot slutten av utviklingsprosessen gjennom en itereringsprosess med tre intervjuer. Vedlegg B viser en introduksjon og hensikt med testingen, prosedyre for utføring av brukertestingen hver deltaker ble introdusert med. Vedlegge inneholder også et avtaleskjema hver deltaker måtte underskrive før gjennomføring av brukertestingen.

De neste delkapitlene går gjennom hovedpunkter ved intervjuene. De viktigste tilbakemeldingene og observasjonene fra en samlet vurdering av intervjuene gjengis anonymt i henhold til avtaledokumentet. Gjennomførte scenario og spørsmål ved intervjuene er dokumentert ved vedlegg B.

8.3.1 Iterasjon 1

Ved gjennomføringen av første iterasjon ble alle deltakere introdusert til brukertestingen gjennom en introduksjon av prosjektet, hvilken kontekst det gjennomføres, og deres rolle ved eventuell deltakelse. Avtaledokument ble gjennomgått og skrevet under av samtlige, med innføring i deres rettigheter som deltakere.

Ved gjennomføring av tildelt scenario klarte alle deltakere å navigere webapplikasjonen riktig og gjennomføre scenarioene på relativt kort tid. Under gjennomføringen fikk gruppen tilbakemeldinger som i hovedsak gikk på knappene, og at de ikke alltid føltes som trykkbare ettersom musen ikke endret form til klikkform. Enkelte brukte lengre tid på å identifisere at enkelte knapper var klikkbare. Gruppen fikk også tilbakemelding på lite utfylt informasjon ved enkelte av de ulike sidene i administrator-, kunde- og leverandørsidene. Mottakere ved mailfunksjonaliten virket noe komplisert for enkelte.

Ved refleksjonsspørsmålene fikk gruppen i hovedsak tilbakemelding fra flere om det visuelle, spesielt at det er mye tomrom ved sidene og mye hvitt. Logikken virket bra for alle deltakere, med unntak for noen vanskelige identifiserbare knapper.

8.3.2 Iterasjon 2

Ved andre iterasjon har gruppen arbeidet med tilbakemeldingene fra forrige iterasjon, og implementert forslag på løsninger ved problemene. Henholdsvis gjort knappene lettere å identifisere, fjernet en informasjonsside med notater og lagt

dette til ved generell informasjon om vedkommende for å fylle sidene mer. Gruppen har også endret noe på designet med mailfunksjonaliteten for å gjøre det mer logisk med mottakere.

Ved gjennomføring av scenario klarte alle deltakere igjen å navigere seg og gjennomføre scenarioene på kort tid. Denne gangen fikk deltakerene utdelt to nye scenario, se vedlegg B. Gruppen fikk ingen signifikante tilbakemeldinger om endringer ved systemet gjennom gjennomføringen.

Ved refleksjonsspørsmålene fikk gruppen en tilbakemelding om at alt hovedsaklig så bra ut. Samtidig fikk gruppen igjen tilbakemelding vedrørende utseende av mailfunksjonaliteten. Denne gangen både om sendfunksjonaliteten, men også om oversikt av eksisterende mail. Gruppen fikk også noe tilbemeldinger på mye tomrom, men fikk bekreftet at den implementerte endringen gjorde det bedre.

8.3.3 Iterasjon 3

Ved tredje iterasjon kom gruppen med en ny løsning på designet av mailfunksjonaliteten.

Deltakerene gjennomførte de to nye scenarioene, og alle klarte igjen å navigere scenarioene på kort tid. Under gjennomføringen ble det ikke pekt ut spesifikke potensielle forbedringspotensialer.

Ved spørsmålene etterpå var tilbakemeldingene vage, slik som ved forrige iterasjon. Det ble igjen påpekt en del tomrom, og noe mindre småpirk ved mailfunksjonaliteten igjen. Gruppen brukte spørringen mot slutten for å forsikre at gruppen forsto tilbakemeldingen for videre forbedring.

8.4 Evaluering

Virkningsgraden ved testingen kan ha påvirket sluttresultatet av prosjektet på flere områder. Gruppen har prioritert å teste systemet etter beste evne, og har slik beskrevet ved øvrige delkapitler, implementert ulike testinger for utarbeidingen av prosjektet. Ved de nedre delkapitlene vil virkning, gjennomføring og den generelle prosessen vedrørende testene drøftes.

8.4.1 Integrasjonstesting

Ved implementasjon av integrasjonstestene opplevde utviklerne et par problemer, mest i forhold til at de ikke har skrevet så mange tester, og spesielt ikke for en applikasjon som hadde så mange komponenter og eksterne avhengigheter. Når det ble funnet løsninger på problemene gikk testingen fort framover. Ved bruk av integrasjonstestingen fikk utviklerne muligheten til å se hvordan ny funksjonalitet påvirket applikasjonen og om applikasjonen fortsatt fungerte som forventet. Dette effektiviserte tidsbruken ved validering av ny funksjonalitet. Testingen første også

til at koden ble effektivisert, ettersom den fremhevet noen hull i autoriseringen og som nå er blitt fikset.

Gruppen startet å skrive tester for applikasjonen ganske sent ut i prosjektet. Om gruppen hadde startet tidligere å skrive testene, ville testene mest sannsynlig hatt en større grad av kvalitet og omfang. Det ville vært mulig å bruke Postman for automatiserte tester, men gruppen ønsket å kunne skrive og eksekvere testene i VSCode. Automatiserte tester via Postman er gjort i Postman sin egen applikasjon. Utviklerene kom over Jest, som er mye brukt ved testing av TypeScript og Javascript og passet situasjonen godt.

8.4.2 Unittesting

Ved unittesting av frontend-komponenter utarbeidet gruppen tester for et mindre utvalg komponenter før det ble besluttet at denne testingen ikke gav et godt nok resultat i forhold til den tiden det tok å implementere testene, se 8.2. Gruppen brukte lengre tid på å implementere fungerende tester, og ettersom alle startet prosjektet med tilnærmet ingen erfaring fra webutvikling ble det utført flere endringer ved koden underveis. Dette betydde også at gruppen ble nødt til å endre testene regelmessig dersom de skulle fungere, og denne prosessen tok så lang tid at gruppen ble nødt til å prioritere bort funksjonalitet dersom de skulle implementeres ordentlig. Gruppen kunne lagt større vekt på implementasjon av testingen tidligere, men ettersom det ble gjort regelmessige refaktorering av koden ble det vanskelig å gjennomføre. Samtidig som det var knapt med tid for gjennomføringen, ble det også dratt lite nytte ut av de testene som allerede var implementert. Gruppen bestemte derfor å ikke prioritere unit-testing av frontend-komponenter.

8.4.3 Brukertesting

Brukertestingen av systemet gav gruppen et innblikk i hvordan en tilfeldig person ville håndtere systemet slik det er implementert, se 8.3 for beskrivelse av gjennomføring. Hovedsaklig fikk gruppen gode resultater fra testingen ved at alle scenarioer ble gjennomørt uten veiledning fra gruppen. Gruppen fikk en overordnet bekreftelse på at systemet er mulig å navigere uten spesielle forkunnskaper. Samtidig fikk gruppen et unikt innblikk ved deltakerene sine tanker rundt navigasjonen, som er tatt med videre ved endringer.

Tilbakemeldingene vedrørende forbedringspotensialer fremsto noe uklart, og enkelte deltakere hadde ingen spesielle forbedringspunkter. Samtidig fikk gruppen tilbakemeldinger om mye tomrom, dårlig identifiserbare knapper og en noe uoversiktlig mailfunksjonalitet. Gruppen fikk gode tilbakemeldinger fra de endringene som ble utført, men mailfunksjonaliteten forble noe uoversiktlig, selv ved siste iterasjon. Ettersom gruppen bare satt av tid til tre iterasjoner fikk ikke gruppen mulighet til å teste siste implementerte mailfunksjonalitet ved brukertestingen.

Dersom gruppen skulle utarbeidet testprosessen på en annen måte ville det blitt

prioritert å utføre testingen på et tidligere stadium slik at det ble lagt av tid til eventuelle ekstra iterasjoner dersom det var ønskelig. Gruppen fikk kun gjennomført testingen ved et mindre utvalg deltakere. I ettertid ville gruppen prioritert å legge av mer tid til rekrutteringen av et bredere og noe større utvalg deltakere, selv om brukergruppen som utførte testingen også var av et bredt utvalg.

Kapittel 9

Diskusjon

For å gjennomføre et prosjektarbeid i den grad gruppen har lagt opp til fra planleggingen kreves det store mengder forberedelser, kontinuerlig beslutninger om viktige valg, og en arbeidsprosess med tydelige krav og delmål. Gruppen har møtt på utfordringer underveis som har påvirket det endelige resultatet av prosjektet. Gjennom dette diskusjonskapittelet vil gruppen drøfte utfordringer, positive synspunkter, hva som har blitt lært, og hva gruppen eventuelt ville gjort annerledes til neste utførelse vedrørende både utvikling- og planleggingsprosessen. Det er allerede utført drøftinger ved tidligere aspekter gjennom rapporten, og disse vil bli referert til.

9.1 Gjennomføring

For gjennomføringen av prosjektet forholdt gruppen seg til den bestemte utviklingsmodellen, beskrevet i 3. Det har fra oppstartsfasen av prosjektet blitt fremmet et delt ønske om å gjennomføre prosjektet tilnærmet likt utførelsen av reelle prosjekter. Gruppen ønsket en slik gjennomføring for å opparbeide erfaring fra en tilnærmet reell arbeidssituasjon. For å velge en passende utviklingsmodell drøftet gruppen dette før oppstart, se 3.1.2. Gruppen brukte samtidig tid på å drøfte lengden ved utviklingsmodellen, se 3.1.3. Samtidig har gruppen ved gjennomføringen hatt kontinuerlige møter med oppdragsgiver, veileder, evaluerings- og planleggingsmøter, og daglige møter, se G for referater, for å sikre kontinuerlig fremgang i henhold til den definerte kravspesifikasjonen, se kapittel 2.

9.1.1 Utviklingsmodell

For gjennomførelsen av prosjektet benyttet gruppen Scrum-metodikk og tok i bruk én uke lange sprinter, som drøftet ved 3.1.3. I ettertid av gjennomførelsen kan gruppen se tilbake på en godt organisert gjennomførelse av prosjektet. Det som har blitt planlagt har hovedsakelig blitt kontinuerlig gjennomført. Ved valg av ut-

viklingsmodell satte gruppen opp diverse kriterierer som burde oppfylles ved valg av passende utviklingsmodell, 3.1.1. Disse kriteriene la tilrette for en smidig utviklingsmetodikk, hvor Scrum var best tilpasset prosjektsituasjonen, og gruppen er godt fornøyd med dette valget også etter utført prosjekt. For utførelse av tilsvarende prosjekt har gruppen vurdert det slik at, utifra drøftingen ved 3.1.3, Scrum ville vært det beste valget av utviklingsmodell dersom en tilsvarende prosjektsituasjon var tillfelle. Det ble utført endringer underveis der oppdragsgiver ønsket det, og gruppen hadde også egne forslag til endringer ved spesifikasjonen. Dette betyr hovedsaklig at en smidig løsning har vært passende, og ut ifra det andre definerte kriteriene om tid og mulighet til å følge utviklingsprosessen gjennom delmål, har drøftingen av utviklingsmodell ved 3.1.3 gitt resultat i form av en passende utviklingsmetodikk.

Samtidig er det i ettertid mulig å se tilbake på enkelte bestemmelser som kunne blitt gjort annerledes. De en uke lange sprintene betyr at gruppen har planlagt ukentlige sprintmøter, for evaluring og planlegging, hver onsdag gjennom hele prosjektperioden, se vedlegg G. Valget vedrørende onsdagsmøter ble hovedsakelig besluttet ettersom oppdragsgiver kunne sette av grupperom for ukentlig arbeid ved deres kontorer på onsdager. Etter diskusjon i gruppen ble dette derfor sett på som en naturlig dag å møtes fysisk i løpet av uken, og dermed utføre planleggings- og evalueringsmøter i henhold til Scrum-metodikken. I ettertid kan gruppen se tilbake utførelsen av sprintmøtene på onsdager som mindre passende. Dette førte til et ufrivillig avbrekk fra arbeidet ved helgene midt i sprintperioden. Ved å ha to arbeidsdager før helg, og to arbeidsdager etter helgen har det vært noe vanskelig å utføre sprintarbeidet med god arbeidsflyt. Grunnet mye ny metodikk og teknologi ved utviklingsprosessen har det heller ikke vært ideelt å ha disse avbrekkene ettersom det kan ta tid å komme inn i samme tankesett og arbeidsflyt. Dersom gruppen hadde utført prosjektet på nytt ville evaluering og planlegging av spring bli plassert ved mandag eller fredag.

Gruppen har som nevnt valgt å utføre oppgaven ved én uke lange sprinter. Som diskutert ved 3.1.3 har valget om en ukers sprinter blitt vurdert etter at gruppen mente regelmessige møter ville være foretrukket dersom noe ikke skulle gå som planlagt. Etter utførelsen er gruppen godt fornøyd med denne beslutningen. De en uke lange sprintene har også sørget for at gruppen har vært kontinuerlig nødt til å innlevere arbeid, og dersom en sprint blir ferdig utført er det mulig å ta grep ved neste sprintplanleggingsmøte uten for store konsekvenser. Gruppen har ved enkelte sprinter hatt problemer med å gjennomføre sprintmål grunnet sykdom, dårlige estimeringer, og dårlige beskrevede oppgaver. Gruppen hadde en sykdomsperiode som rammet ved sprint ni, se vedlegg G. Ettersom gruppen aldri har gjennomført en slik arbeidsprosess føler gruppen at de gjorde riktig i å legge tilrette for at slike hendelser mest sannsynlig ville oppstå.

Dersom gruppen hadde tatt i bruk større sprintperioder ville sykdomsperioden gruppen opplevde i løpet av prosjektperioden rammet hardere ettersom det ikke hadde vært mulig å justere og redelegere arbeidsoppgaver etter en uke. Sam-

tidig ville lengre sprintperioder gjort det vanskelig å komme seg inn i Scrum-metodikken. Ved å ha gjennomført én uke lange sprinter har gruppen opparbeidet dobbel så mye erfaring med estimering- og planleggingsprosesser enn dersom de hadde valgt to uker.

9.1.2 Estimering

Gruppen har benyttet estimeringsmetoden *planning poker*, som diskutert ved 3.2.3 gjennom hele prosjektet. Estimeringen har vært varierende i forhold til faktisk tid, men gjennom kontinuerlige estimeringer har gruppen opparbeidet erfaring som har ført til mer presise estimeringer underveis. Som diskutert ved 9.1.1 har én uke lange spriner gjort det mulig å feile, og bli bedre på estimeringen gjennom prosjektperioden.

I tillegg til at gruppen ikke har mye tidligere erfaring ved estimeringsmetodikk, har gruppen også aldri utviklet et prosjekt av denne størrelsen, og har også benyttet nye metoder for utarbeidingen av resultatet. For å gjennomføre prosjektet, slik at forventningene ved spesifiserte mål og kravspesifikasjon blir oppfylt, har gruppen måttet fordele arbeidsområder mellom seg. Det har hovedsaklig blitt fordelt slik at to jobber med backend-utvikling, og to jobber med frontend-utvikling. Ved estimering av backend- og frontend-spesifiserte deloppgaver har gruppen vært delt i hvor mye som skal til for å løse deloppgavene. Dette vil si at estimering av disse deloppgavene har vært vanskelig å få til, ettersom kun to av gruppemedlemmene har erfaring med arbeidet og arbeidsmengden ved utførelsen. Samtidig har gruppen basert backlogen på use caser, og flere av disse har ingen tydelig ferdigstillelse og må derfor deles opp i mindre deloppgave, som har vært noe utfordrende underveis. Dette har gjort estimeringsøktene vanskeligere, men dette er også noe gruppen ble bedre på underveis.

Dersom det har oppstått uenigheter har gruppen begrunnet sitt syn på saken. Etter disse diskusjonene har det hovedsaklig ført til enighet, men dersom dette ikke har vært tilfelle har Scrum-master tredd inn med en ekstra stemme, slik bestemt i 1.10. Denne metoden for å løse avstemningene har fungert effektivt, og gruppen har unngått større uenigheter.

Andre estimeringsmetoder som kunne fungert for gruppen er blant annet *affinty mapping*[1] eller *T-shirt sizes*[2]. Disse var aktuelle ettersom de er estimeringsmetoder som gir estimater i størrelser som er mer vagt definert, for eksempel ”ekstra liten” eller ”medium” arbeidsmengde. Gruppen valgte derimot å ikke bruke disse estimeringsmetodene siden gruppemedlemmene følte at de ble bedre til å estimere ved bruk av planning poker.

[1] `https://www.greycampus.com/opencampus/agile-certified-practitioner/affinity-estimation#:~:text=Affinity%20Estimating%20is%20a%20techniquein%20preparation%20for%20release%20planning`

[2] `https://www.redagile.com/post/tshirt-sizing`

Overordnet vil gruppen kunne vise til stor forbedring av estimeringsferdigheter gjennom prosjektperioden. Ved å bruke planning poker, og kjøre estimeringsmøter ved sprintplanleggingsmøtene tar gruppemedlemene med seg verdifull erfaring vedrørende estimering, men dette var også en mulighet å diskutere og sette tydelige krav til gruppemedlemene gjennom prosjektet. Gruppen er fornøyd med valget av planing poker som estimeringsteknikk.

9.1.3 Møter

Gruppen har planlagt og gjennomført kontinuerlige møter med oppdragsgiver, veileder, og innad i grupen gjennom hele prosjektperioden.

Ved møtene med oppdragsgiver og veileder ble det hovedsaklig satt av tid til et ukentlig møte for diskusjoner, slik beskrevet i 3.2.5 og 3.2.6. De ukentlige møtene med oppdragsgiver og veileder har blitt gjennomført dersom noen av partene hadde noe å ta opp, og ved oppstart ble dette tilbudet hyppig benyttet. Senere i prosjektperioden ble møter med veileder hovedsaklig gjennomført ved sluttfasen og rapportskrivingsfasen. På samme måte ble møtene med oppdragsgiver gjennomført ved utviklingsfasen, og ved rapportskrivingsprosessen ble disse sjeldnere gjennomført. Gjennomførelsen av møter med oppdragsgiver og veileder har blitt gjennomført etter behov, og ved god kontinuerlig kommunikasjon mellom alle ledd har gruppen gjennomført en god møteprosess ved hele prosjektperioden. Gjennom en ordentlig gjennomførelse av møter med både oppdragsgiver og veileder har gruppen tatt med seg verdifull læring ved møteplanlegging og gjennomføring.

Innad i gruppen ble det arrangert Scrum møter hver onsdag, som beskrevet ved 9.1.1. Det ble gjennomført Scrum-møte etter planlagt gjennomføring ved alle ukene. Alle gruppemedlemene har deltatt på samtlige møter, med unntak av noe sykdomsfrafall, se sprint ni ved G. Samtidig har gruppen utført regelmessige daglige møter på arbeidsdager. Disse møtene har hovedsaklig bare omhandlet hvilke oppgaver hvert gruppemedlem jobber med, og hvordan sprinten går. Ved å kontinuerlig gjennomføre disse møtene har gruppen opparbeidet en god læring vedrørende utførelse av utviklingsmodellen Scrum ved et reellt prosjekt.

9.2 Tekniske aspekter

Gruppen har brukt god tid til diskusjon vedrørende implementeringsmetodikk av oppgaven. Ved valg av teknologier, arkitekturer, dokumentasjonsmetodikk, og gjennomførelsen vedrørende de tekniske aspektene har gruppen opparbeidet bred erfaring fra mange nye teknologier og metodikker.

9.2.1 Systemarkitektur

Istedenfor å implementere en serverless struktur kunne gruppen utviklet systemet etter tjenesten Azure Web App[3]. Azure Web App er en PaaS som gir utviklere muligheten til å lage web applikasjoner, og REST API-er som kan kjøres i skyen eller lokalt avhengig av behov.

Dersom gruppen hadde valgt å implementere Azure Web App istedenfor Azure Functions ville enkelte ting ved applikasjonen sett annerledes ut. For eksempel så har ikke Azure Web App de samme problemene med cold start, se 5.2.2, som Azure Functions har. Azure Web App vil kjøre konstant, og vil derfor ikke ha noen cold start, utenom når applikasjonen blir startet.

Som tidligere nevnt, er komponentene/funksjonene i applikasjonen løst koblet sammen og dette er en konsekvens av serverless arkitektur. At komponentene er løst koblet sammen gjør at det er enkelt å legge inn nye komponenter. Dette passer godt sammen med utviklingsmodellen vi valgte, som er tilrettelagt for forandringer i krav og funksjonalitet for applikasjonen. Azure Web App følger en monolittisk struktur som fører til at komponentene er tettere knyttet opp mot hverandre enn i serverless arkitektur[4]. Om komponentene er tett knyttet sammen, vil det vanskeligere å implementere nye komponenter.

Kostnadsplanen som Azure Functions applikasjonen kjører på er en *Consumption* plan der man betaler etter hvor mye man bruker. Et alternativ til denne planen ville vært å velge en plan som *App Service*. Med en App Service plan så betaler man en konstant mengde, uansett hvor mye eller lite man bruker av tilgjengelig ressurser. Ettersom applikasjonen er laget til TietoEvry Brumunddalen sin situasjon, vil det ikke være verdt kostnadene å bruke en App Service plan. Men om TietoEvry velger å utvide brukerbasen til applikasjonen i stor grad, kan de være mer lønnsomt å bruke en App Service plan.

Når gruppen så på de ulike funksjonalitetene som applikasjonen skulle tilby, kom gruppen fram til at funksjonene som skal levere disse funksjonalitetene, ikke ville være aktive over lang tid og vil heller ikke kreve store mengder med minne. App Service plan er kostnadseffektiv når det er funksjoner som er aktive over lengre perioder og krever mye minne. Dette er på grunn av at App Service er en statisk kostnadplan, så selv om funksjonene vil kjøre over lengre perioder vil ikke dette påvirke kostnadene. Derimot så passer Consumption planen godt til situasjoner der funksjonene ikke varer så lenge og heller ikke bruker mye minne. For å se hvordan kostnaden til Consumption planen er regnet ut, se 5.2.1. I tillegg så vil funksjoner eksekvert under en Consumption plan bli kansellert etter fem minutter[5]. Om TietoEvry ønsker å utvide funksjonalitetene med funksjonalitet som

[3] `https://docs.microsoft.com/en-us/azure/app-service/overview`

[4] `https://docs.microsoft.com/en-us/dotnet/architecture/containerized-lifecycle/design-develop-containerized-apps/monolithic-applications`

[5] `https://docs.microsoft.com/en-us/azure/azure-functions/functions-scale#timeout`

krever funksjonene som er aktive over lang tid, så kan det være mer lønnsomt å bruke en App Service plan.

Gruppen har tidligere drøftet valg vedrørende database og autentisering i 5.5 og 5.4.

9.2.2 Teknologier

Ved valg av teknologier har gruppen implementert en rekke teknologier og metoder for utviklingen av systemet. Ved 6.1 blir de ulike teknologiene presentert med en kort begrunnelse for valget av løsningene. Flere av disse er demonstrert ved kapittel 6, men enkelte av disse valgene ble sett som større og definerende for utviklingsprosessen.

Som diskutert, ved blant annet 5.6, har gruppen valgt å implementere brukergrensesnitt og danne komponenthieraerki etter det komponentbaserte biblioteket React. Valget av rammeverk/bibliotek for utvikling av ble nøye diskutert ved oppstartsfasen av prosjektet, og det hadde vært mulig å implementere brukergrensesnittet etter flere andre løsninger. Gruppen så på blant annet Angular og Vue.js, men etter lengre diskusjoner kom gruppen frem til at alle metodene virket som gode løsninger for implementasjonen. Gruppen valgte derfor å følge anbefaling fra bekjente om å implementere brukergrensesnittet med React.

Gruppen valgte å benytte TypeScript over JavaScript på grunn av muligheten for å gi interfaces til objekter. Typescript har hjulpet gruppen med å unngå feil, for eksempel å sende med text til en funksjon som krever et nummer. Det har også vært spesielt nyttig i frontend da objektene som ble returnert fra backend ofte innehold mye data. Typescript har også gitt mulighet for autofullføring av variabler som har gjort utviklingen av applikasjonen raskere.

Git har blitt brukt for utgivelse av applikasjonen ved publiseringer til GitHub, se 7.1. Gruppen har også brukt Git for å dele koden med hverandre. Gjennom bruken av Git kunne gruppen brukt ulike grener for utvikling og hovedgren, men ettersom utgivelses pipeline sjekker det nyeste publiserte gjennom testing og builds har ikke gruppen sett dette nødvendig for utviklingsprosessen, 6.4.

Overordnet har valg av teknologier fungert bra. Gruppen hadde ingen tidligere erfaring med webteknologier, og stilte derfor uten foretrukkende valg. I ettertid er gruppen enige om at valgene som er gjort underveis har passet løsningen godt, og ser ingen klare feil ved valgene av teknologier.

9.3 Grafiske aspekter

Gruppen har tatt flere store valg vedrørende de grafiske aspektene ved systemet, se 4. Begrunnelsen for flere av de grafiske valgene blir definert ut fra utvalgte patterns, desgignprinsipper, og lovehjemmelen vedrørende universell utforming,

se 4.2, 4.3 og 4.3. Gruppen har fra oppstartsfasen, og resultatmålet ved 1.6.2 definert et ønske om å utarbeide et oversiktlig og brukervennlig brukergrensesnitt.

9.3.1 Utviklingsprosess

For å sikre utarbeiding av et brukergrensesnitt, som oppfyller de stilte kravene og spesifikasjonene, har gruppen benyttet prototypeutvikling og brukertesting, beskrevet i 4.1. Gruppen startet denne prosessen i oppstartfasen for å danne et bilde av hvordan webapplikasjonen kunne ende opp med å se ut. Den tidlige low fidelity prototypen, beskrevet ved 4.1.1, og klarering fra oppdragsgiver, gjorde at gruppen fikk en klar visualisering av hvilken retning det skulle utvikles. Etter denne klareringen sikret gruppen at alle arbeidet etter en tilnærmet lik løsning, og samtidig sikret gruppen at oppdraggiver var på samme bølgelengde og fulgte samme tankegang ved utførelsen. Ved å unnlate denne prosessen ville det lettere kunne oppstå misforståelser både innad i gruppen, og med oppdragsgiver. Denne prototypen eliminerte ikke muligheten for misforståelser, men gruppen så det som en mulighet for å forhindre større misforståelser senere i prosjektet.

Senere har gruppen implementert en mer avansert prototypemodell i Figma, se 4.1.2. Ved å konstruere brukergrensesnittet med innarbeidet funksjonalitet har gruppen sammen gjort det mulig å visualisere hvilke implementasjoner av webapplikasjonen som burde implementeres. Gjennomføringen av high fidelity prototypene har forenklet utviklingsprosessen, og ved ny gjennomførelse ville dette blitt prioritert på nytt.

Gruppen gjennomførte også brukertesting, som beskrevet ved 8.3, for å sikre et oversiktlig og intuitivt resultat ved webapplikasjonen i henhold til restultatmålet gruppen har satt ved 1.6.2. Gjennomføringen og virkningen av brukertesting blir drøftet ved 8.4.3.

9.3.2 Andre føringer

Gruppen benyttet diverse prinsipper og patterns gjennom utviklingen av det grafiske aspektet ved webapplikasjonen, og samtidig i henhold til kravene om universell utforming.

Designprinsippene, se 4.4, ga gruppen mulighet til å utvikle brukergrensesnittet i henhold til fast dokumentasjon vedrørende allerede eksisterende prinsipper. Gruppen valgte ut et par prinsipper og utviklet det grafiske aspektet i henhold til disse. Den samme prosessen ble utført ved patterns, ved 4.2. Ved å definere prinsipp og patterns tidlig i prosjektfasen har gruppen jobbet mot å løse det grafiske designet i henhold til disse. Gruppen kunne ha inkludert et større utvalg prinsipper og patterns, men hovedsaklig så gruppen de utvalgte retningslinjene som utfyllende nok til å utvikle et godt resultat basert på disse. Ved ny utførelse av tilsvarende prosjekt ville gruppen mest sannsynlig definert flere prinsipper og

patterns, ettersom dette ville gjort det enklere å dokumentere de ulike løsningene i henhold til dokumenterte løsninger.

Kravene om universell utforming har blitt høyt prioritert, ettersom dette er lovpålagt ved utforming av nettsider, se 4.3. Gruppen har derfor fulgt disse rådene i større grad, der det sees relevant, og har utformet webapplikasjonen etter dette.

9.4 Måloppnåelse

Gruppen har fra tidlig i prosjektfasen definert ulike krav for resultatet. Ved 1.6 blir det definert et utvalg effektmål, resultatmål og læringsmål. Effektmålene må vurderes senere for å dokumentere virkningsgraden systemet har hatt. Samtidig har gruppen definert ulike opperasjonelle og sikkerhetskrav for utarbeidingen av systemet.

9.4.1 Resultatmål

Resultatmålet, se 1.6.2, beskriver en webapplikasjon som samler informasjon vedrørende kunder og leverandører på en plattform. Dette har gruppen utarbeidet og fungerer fullt funksjonelt. Som demonstrert i 5.5 har gruppen utarbeidet en database hvor informasjon om alle aktører i webapplikasjonen lagres ved beskrevede relasjoner til hverandre.

Systemet skal også gjøre kunde- og leverandørrelatert informasjon tilgjengelig gjennom et oversiktlig brukergrensesnitt. Gjennom utarbeiding av prototyper for brukergrensesnitt, designprinsipper, patterns og utførelse i henhold til universell utforming, har gruppen arbeidet med et klart fokus om å utarbeide et godt oversiktlig brukergrensesnitt, se 4. Etter utarbeiding av disse metodene har gruppen også benyttet brukertesting for å sikre et fullt fungerende og oversiktlig utarbeidet brukergrensesnitt i henhold til kravet ved resultatmålet, se 8.3.

Ved å sammenlikne det tidlige resultatmålet med utarbeidet resultat har gruppen klart å utarbeide et resultat som tilfredstiler det klarerte resultatmålet.

9.4.2 Læringsmål

Læringsmålene var tidlige utarbeidede mål for prosjektperioden i henhold til læringsmålene definert ved bacheloremnet, se 1.6.3. Overordnet er disse læringmålene fordelt ved prosjekt- og teknologispesifikke læringmål.

De prosjektspesifikke læringsmålene er alle utført, med lære å estimere har vært noe trøblete, diskutert ved 9.1.2. Uten om dette har gruppen kontinuerlig brukt Scrum-metodikk, holdt kommunikasjon med oppdragsgiver gjennom møter og opprettet kanal ved Microsoft Teams og dokumentert prosessen, spesielt tydelig ved vedlegg G og I, og henvist til kilder underveis.

De teknologiske læringsmålene er i større grad utfylt på same måte. Gruppen har benyttet git gjennom hele utviklingsfasen, diskutert ved 9.2.2, anvendt de bestemte teknologiene og tjenestene, demonstrert gjennom kapittel 6, og jobbet for å designe en brukervennlig og intuitiv webapplikasjon, diskutert i 9.3. Ved testing, som diskutert i kapittel 8, har gruppen møtt på noen problemer ved utført testing i utviklingsprosessen, men har benyttet det for utviklingen av systemet. Detaljert drøfting vedrørende testing blir diskutert ved 8.4.

Overordnet vil gruppen si at de spesifiserte læringmålene er gjennomført. Testing ble implementert, og delvis gjennomført, men overordnet er ikke gruppen fult fornøyde med resultatet av denne prosessen. Brukertestingen og de implementerte integrasjonstestene fungerte bra, men ettersom unittesting ikke ble skikkelig implementert ved frontend-komponentene er ikke gruppen fult fornøyd med hva de fikk til med testingen. Allikevel føler gruppen at alle læringsmålene er oppfylt, ettersom gruppen har klart å implementere testing.

9.4.3 Operasjonelle- og sikkerhetskrav

De definerte operasjonelle og sikkerhetskravene er dokumentert ved 2.4 og 2.5 er definert på et senere stadium i løpet av rapporten. Disse er alle utført, og det blir henvist til stedene disse utføres ved kapittelene.

9.5 Videre utvikling

Utover minimumsløsningen, hadde oppdragsgiver forslag til noen utvidelser dersom det ble tid. En av disse utvidelse var å legge til nye informasjonsfelter på kunder ut i fra hva slags kategorier de tilhører. Funksjonalitet for dette er lagt til i backend, men det ble ikke tid til å legge det til i frontend.

Fordi applikasjonens backend er implementert som flere, løst koblede Azure Functions 5.2.3, kan backenden relativt enkelt utvides med ny funksjonalitet. Dette kan gjøres med lav risiko for at det vil kunne påvirke andre deler av applikasjonen på en negativ måte, så lenge denne nye funksjonaliteten behandler data i databasen på riktig måte, se 5.6.

For frontenden er både koden og designet på siden godt egnet for videre utvidelser. React, som er brukt for å lage frontend, legger opp til at nettsidens kode bygges opp som en samling av moduler, se 6.6.2. Dette modulære designet gjør det relativt enkelt å implementere ny funksjonalitet senere i form av nye moduler. Nettsiden er også designet slik at det er mulig å legge til utvidelser senere, uten at det gjør siden vanskeligere å bruke. Navigering i nettsiden gjøres hovedsakelig gjennom et statisk navigasjonspanel og en sidemeny. Ny funksjonalitet kan legges til som nye knapper på disse menyene, og på den måten unngå å forstyrre brukerens oppfattelse og opplevelse av systemet. En faktor som begrenser videre utvikling av frontenden, er at den er laget som en statisk nettside. Dette vil ha en negativ

påvirkning på i hvilken grad nye utvidelser kan tilpasses hver enkelt bruker[6].

Applikasjonens backenden er laget som en RESTful API, se 6.2.2, og avskilt fra frontenden i Azure. Det vil derfor være mulig å bruke den i andre bruksområder enn nettsiden vi har laget. Dersom det for eksempel er interesse for å lage en komplementær mobilapplikasjon senere, vil det være mulig å bruke backenden til dette.

[6] `https://web.archive.org/web/20190320233700/https://smallbusiness.chron.com/difference-between-dynamic-static-pages-69951.html`

Kapittel 10

Konklusjon

For å konkludere prosjektoppgaven har gruppen utarbeidet en webapplikasjon som tilfredsstiller de utarbeidede målene, ved 1.6. Dette er mer detaljert diskutert ved 9.4. Gruppen har også gjennomført oppgaven i henhold til den spesifiserte utviklingsprosessen, se 9.1.

Systemet fungerer som et CRM-system hvor ulike brukerroller har tilgang til tilpasset funksjonalitet og informasjon. Det ferdigstilte resultatet er presentert for oppdragsgiver, og har blitt godkjent i henhold til de kravene som ble satt for systemet.

Gruppen er fornøyd med resultatet, og føler de har svart godt på oppgaven. De tar med seg mye ny kunnskap, og har opparbeidet en erfaring i en arbeidsrelevant situasjon.

Bibliografi

[1] S. Balaji og M. S. Murugaiyan, «Waterfall vs. V-Model vs. Agile: A comparative study on SDLC,» *International Journal of Information Technology and Business Management*, årg. 2, nr. 1, s. 26–30, 2012.

[2] P. Abrahamsson, O. Salo, J. Ronkainen og J. Warsta, «Agile software development methods: Review and analysis,» *arXiv preprint arXiv:1709.08439*, 2017.

[3] D. Rawsthorne, «Sprint Length: What Length is the Right Length?,» 2019. adresse: https://3back.com/scrum-tips/sprint-length-what-length-is-the-right-length/#:~:text=It%5C%27s%5C%20a%5C%20rule%5C%20of%5C%20Scrum%20Story%5C%20and%5C%20get%5C%20it%5C%20Done.

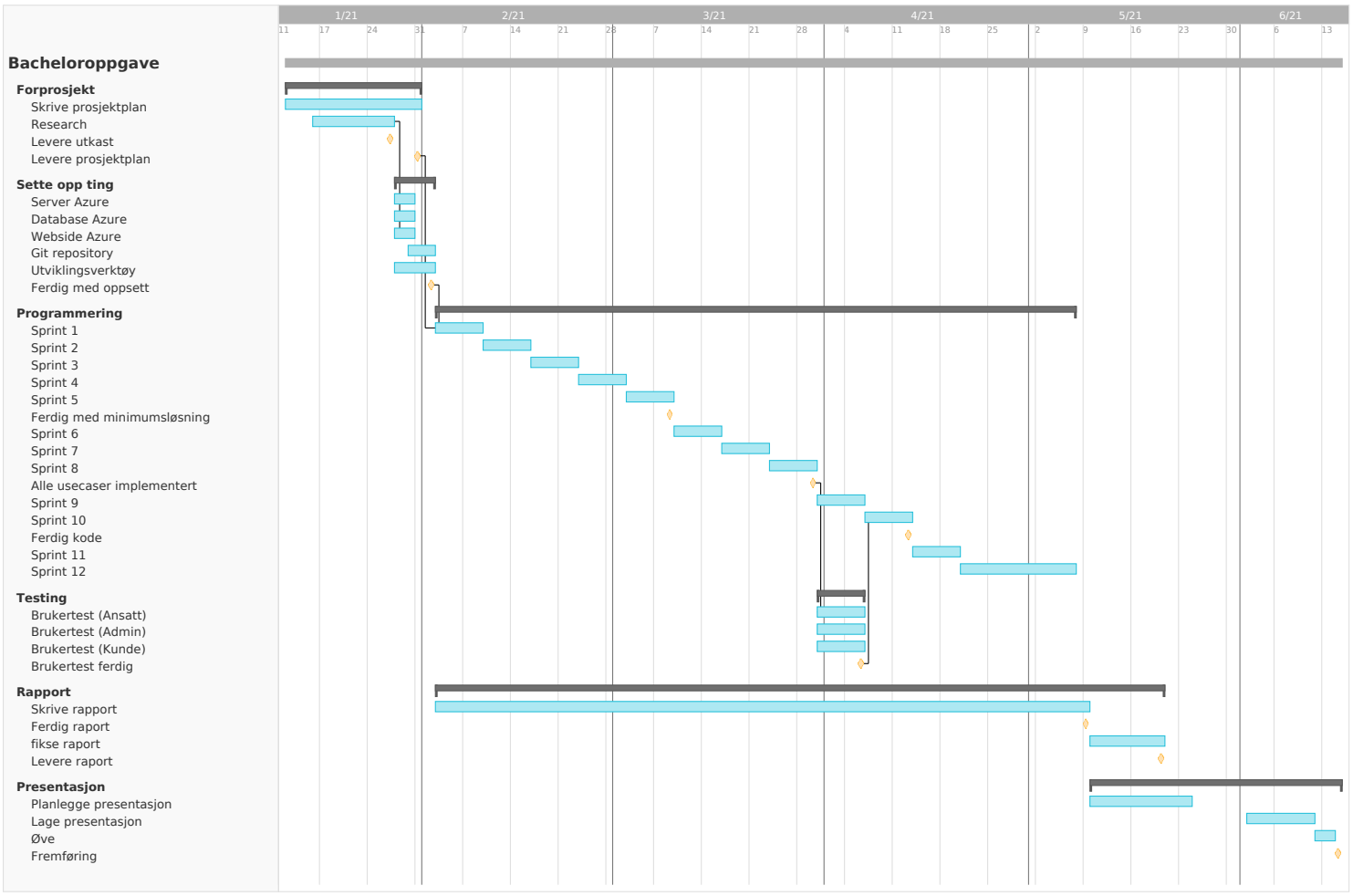
[4] I. Boroujen, «A case study research on software cost estimation using experts' estimates, wideband delphi, and planning poker technique,» *International Journal of Software Engineering and its applications*, årg. 8, nr. 11, s. 173–182, 2014.

[5] M. Cohn, *Agile Estimating and Planning*. Upper Saddle River, NJ, USA: Prentice Hall, 2005.

[6] «WCAG 2.0-standarden,» adresse: https://www.uutilsynet.no/wcag-standarden/wcag-20-standarden/86?field_list_taxonomy_one_target_id=All.

[7] «Forskrift om universell utforming av informasjons- og kommunikasjonsteknologiske (IKT)-løsninger,» 2013. adresse: https://lovdata.no/dokument/SF/forskrift/2013-06-21-732.

[8] «Serverless apps: Architecture, patterns, and Azure implementation,» 2020. adresse: https://docs.microsoft.com/en-us/dotnet/architecture/serverless/.



Vedlegg A

Statusrapporter

Statusrapport 17.02.2021

1. **Status for punktene under:**

   Hva er avsluttet/under arbeid? Er tidsfrister: overholdt / overskredet / kritiske for:

   (a) Planlegging (jfr. fremdriftsplan)

   Planlegging har gått greit, vi har gjort mesteparten av det de oppgavene vi har planlagt. Vi følger scrum og de oppgavene vi ikke har rukket har blitt med over i neste sprint.

   Vi har satt opp faste møttider mandag, tirsdag, onsdag og fredag. Onsdag er vi hos TietoEvry, der vi startet sprintene med sprint planing meeting. Tirsdag har vi møte med veileder. Mandag og Fredag har vi fysiske møter der vi jobber med oppgaven.

   (b) Organisering av gruppens arbeid og ansvarsområder

   Gruppa er organisert i to grupper, med to medlemmer hver. Der den ene har ansvar for frontend og den andre gruppa har ansvar for backend. Vi har også valgt en scrummaster og en prosjektleder.

   (c) Klargjøring av problemstilling (kravspek'en)

   Vi har laget Use Caser, definert diverse operasjonelle- og sikkerhetskrav med arbeidsgiver i forbindelse med kravspesifikasjon. Vi har kommet godt på vei og har alt vi trenger, mangler en siste finpuss og å legge dette inn i bacheloroppgaven.

   (d) Løsningsmetode

   For å løse oppgaven har vi valgt å jobbe etter en smidig metode (Scrum). Vi gjør dette fordi kravspesifikasjonen kan endre seg.

   Vi har valgt å bruke TypeScript (JavaScript) som programmeringsspråk fordi det kan brukes både frontend og backend. På frontend bruker vi rammeverket React og vi bruker NodeJS for dataoverføring til og fra backend med RestAPI. I databasen bruker vi MongoDB. Både databasen og nettsiden blir levert av Microsoft Azure.

   (e) Rapportskriving

   Til nå har vi kommet godt i gang med rapportskriving. Vi har jobbet godt med prosjektskissen, kravspesifikasjonen, og i tillegg satt opp rapporten for den endelige innleveringen.

2. **Totalstatus for punktene over:**

   Totalstatus for punktene over er at vi har kommet godt i gang med prosjektet og er fornøyde med slik arbeidet har fungert de første ukene.

3. **Muligheter. Trusler/Problemer:**

   Mulige trusler mot prosjektet kan være at medlemmer blir syke. Dette opplevde vi for en uke siden og det syke medlemmet rakk ikke å fullføre sine arbeidsoppgaver. Andre mulige trusler er uenighet, manglende kunnskap og dårlig planlegging av tid.

4. **Hva med motivasjon:**

   (a) Gruppens samarbeid, arbeidsformer og organisering.

   Gruppa har et godt samarbeid og motivasjonen for prosjektet er høy. Vi har fått jobbet mye ved fysisk tilstedeværelse, og dette gjør det mye lettere å diskutere og samarbeide enn ved bare digitale møter.

   (b) Forhold som oppmuntrer eller frustrerer.

   Det oppmuntrer å se at arbeidet blir gjort til riktig tid og at sprintene fungerer. Vi har tydelige ukeplaner som gjør arbeidet forutsigbart og lett å holde oversikt. Dette er med på å skape et oppmuntrende arbeidsmiljø hvor man hele veien kan få hjelp eller innspill ved behov.

   Noe av det som kan ha frustrert har vært oppgavedelegeringen ved sprint slik at det blir delt ut nok arbeid, og samtidig ikke for mye. Vi har blitt bedre på å estimere tid på oppgaver og bruker planning poker som verktøy.

5. **Hvordan oppleves veileder- og oppdragsgiverkontakten**:

   (a) Fungerer den

   Oppdragsgiver er imøtekommende, tilstede og svarer raskt på spørsmål om det er noe vi lurer på. Oppdragsgiver gir også tilbakemelding når vi har kommet med ideer.

   Selv om veileder ikke kan hjelpe med tekniske utfordringer og spørsmål knyttet til prosjektet, er han veldig til hjelpsom når det kommer til rapportskriving, tips til gode rutiner og andre forhold rundt oppgaven.

   (b) Fornøyd / misfornøyd

   Vi er fornøyde med både oppdragsgiver og veileder.

Statusrapport 07.04.2021

1. **Status for punktene under:**

   Hva er avsluttet/under arbeid? Er tidsfrister: overholdt / overskredet / kritiske for:

   (a) Planlegging (jfr. fremdriftsplan)

   Vi har fortsatt med de samme møtetidene vi hadde ved forrige statusrapport, men på grunn av korona-situasjonen har vi ikke vært på Brumunddal eller møttes fysisk de siste ukene. Vi valgte å ikke møtes i påsken.

   I forhold til fremdriftsplanen vi lagde i prosjektskissen ligger vi en uke eller to bak skjema når det kommer til kodedelen.

   (b) Organisering av gruppens arbeid og ansvarsområder

   Gruppen har fortsatt hovedsakelig to grupper med to medlemmer hver, hvor en gruppe har ansvar for frontend og den andre for backend. I tillegg så har vi lagt opp til at alle gruppemedlemmer skal begynne å skrive på rapporten og har delt ut kapitler basert på medlemmenes preferanser og hva de har jobbet med. Scrum meetings blir fortsatt holdt hver onsdag for å fordele oppgaver, samt diskusjon av nye potensiale oppgaver.

   (c) Ferdigstilt arbeid

   Mesteparten av funksjonalitet har blitt implementert, men det mangler noe før vi skal være ferdig i løpet av to uker. Backend fungerer bra, og det er opprettet et fungerende API hvor frontend har tilgang til en del av funksjonaliteten.

   Alt i alt føler vi at vi ligger helt ok til tidsmessig. Kunne ligget litt bedre an, men vi har mesteparten av hovedfunksjonaliteten implementert.

   (d) Manglende arbeid

   Mangler litt funksjonalitet i applikasjonen, men det er estimert at det skal bli ferdig i løpet av uken. Designet av applikasjonen må også ferdigstilles. Det skal også lages og utføres tester om en eller to uker.

   (e) Rapportskriving

   Vi har i løpet av den siste tiden begynt på rapporten. Ved nåværende tidspunkt jobber vi med å ferdigstille et utkast av fem kapitler i løpet av dagen, 07.04.2021. Vi har brukt påsken på å starte ordentlig med skrivingen.

Vi føler vi har god kontroll på selve rapporten, og er i rute slik vi har planlagt å være.

2. **Totalstatus for punktene over:**

   Totalstatus for punktene over er at vi har kommet langt på vei, men mangler litt for å kunne ferdigstille applikasjonen. Vi føler vi har litt dårlig tid på programmeringen i forhold til gantt-diagrammet. Vi la inn en buffer i diagrammet, så vi har troen på at det skal kunne utføres i god tid før fristen.

3. **Muligheter. Trusler/Problemer:**

   Noen av truslene for prosjektet ved nåværende tidspunkt er relatert til tidsfrister. Det kan være at vi møter på problemer som tar ekstra lang tid, og vi ønsker å være forberedt på dette ved å ligge godt an.

   Til nå har ingen av truslene vi ramset opp sist utløst noen større problemer, utenom å ferdigstille arbeidsoppgavene sine til hver sprint. Noen av oppgavene har tatt lengre tid enn først estimert, og andre oppgaver har på grunn av det blitt utsatt. Mangel på tid er uten tvil det største problemet.

4. **Hva med motivasjon:**

   (a) Gruppens samarbeid, arbeidsformer og organisering.

   Gruppa har et godt samarbeid og motivasjonen for prosjektet er høy. Selv om vi har måttet flytte mange av møtene over til digitale plattformer, har samarbeidet og kommunikasjonen fortsatt å fungere bra.

   (b) Forhold som oppmuntrer eller frustrerer.

   Det er flere forhold som oppmuntrerer på dette stadiet i prosjektet. Vi begynner å se slutten av prosjektet, og vi ser hvordan vi ønsker at applikasjonen skal se ut. Vi jobber hardt alle sammen, og det oppmuntrer at gruppen jobber godt sammen og har god kommunikasjon.

5. **Hvordan oppleves veileder- og oppdragsgiverkontakten**:

   (a) Fungerer den

   Arbeidsgiver har vært gode. Vi har regelmessig møtt opp på deres avdeling i Brumunddal for å jobbe sammen der, og der har det også vært mulig å konsultere om hvor vi ligger i prosessen og om ting er gjort på riktig måte. De har også vært tilgjengelig over teams hele tiden, og har tilbydd seg å hjelpe til dersom vi skulle stå fast noen steder.

   Antall møter med veileder siden forrige statusrapport har ikke vært så mange ettersom gruppen har fokusert mest på programmeringsdelen av prosjektet, men vi har fortsatt satt opp ett møte i uken dersom

det skulle være noen spørsmål. Veileder er tilgjengelig og fungerer fortsatt veldig bra.

(b) Fornøyd / misfornøyd

Vi er fortsatt fornøyde med både oppdragsgiver og veileder.

Vedlegg B

Dokumentasjon av brukertesting

Brukertesting av system.

Introduksjon.

Ved gjennomføring av brukertesting har gruppen lagt vekt på testing av det grafiske aspektet ved systemet. Målsettingen for utviklingsprosessen av systemet er å utvikle et oversiktlig og organisert system, og ved denne testingsprosessen ønsker gruppen å dokumentere eventuelle forbedringsområder. Testingen gjennomføres ved å delegere ulike scenarioer til deltakerne hvor de uten veiledning skal navigere og gjennomfør aktuelt scenario. Deltakerne vil bli bedt om å tenke høyt under utførelsen av scenarioet, og det vil bli stilt noen spørsmål vedrørende deres opplevelse ved systemet.

Testingen blir gjennomført ved et tilfeldig utvalg deltakere med ulik alder, kjønn og erfaring ved liknende CRM-system. Gruppen håper med dette å tilegne seg verdifull innsikt fra ulike deltakere.

Hensikt.

Gruppen har utviklet systemet med et oppsett og brukergrensesnittet som virker både logisk og oversiktlig etter gruppens tankegang. Ved brukertesting skal vi undersøke om et tilfeldig utvalg deltakere på samme måte synes systemet virker oversiktlig og logisk.

Hensikten ved brukertestingen er å finne eventuelle forbedringspotensialer ved systemet for å forsikre at systemet fremstår slik målsettingen ved prosjektet tilsier.

Prosedyre av gjennomføring av brukertesting.

1. Deltaker signerer avtaledokumentasjon om gjennomførelsen av testingen, og at vi kan lagre tilbakemeldinger for å evaluere systemet i en gitt tidsperiode, vedlegg A.
2. Deltaker får tildelt informasjonsdokument for informasjon om gjennomføring av testing, og får også forklart gjennomføringsprosess av intervjuer, vedlegg A.
3. Deltaker får tildelt to scenarioer for gjennomføring, og skal tenke høyt ved hvert steg, vedlegg B.
4. Deltaker blir stilt noen refleksjonsspørsmål for å dokumentere deres opplevelse av systemet, vedlegg B.

Vedlegg A.

Avtaledokument.

Ved deltakelse i denne brukertestingen vil vi dokumentere og benytte oss av tilbakemeldingene ved gjennomførelsen for å evaluere oversiktlighet, logikk og mulige forbedringspotensialer ved brukergrensesnittet av systemet. Deltakelsen er frivillig og opp til hver enkelt om det anses ønskelig å være deltaker ved testingen. Gjennomføringen av testingen vil ta 15-20 minutter.

Gjennomføringen av testing skjer ved et intervju. Deltaker får tildelt et utvalg scenarioer de skal løse ved å navigere systemet, og samtidig snakke høyt hva de tenker ved hvert steg i prosessen for å gjennomføre scenarioet. Etter dette vil deltaker spørres ved noen refleksjonsspørsmål vedrørende systemet.

Ved gjennomføringen av testingen vil intervjuer dokumentere gjennomførelsen og refleksjoner ved spørsmål. Denne dataen vil lagres for senere evaluering av programmet, og deretter slettes med all relevant data knyttet opp mot aktuell deltaker nøyaktig to uker etter intervjuprosess. Lagret data vil inneholde deltakers navn for å identifisere data slik at det kan slettes ved eventuell trekking fra prosess.

Dersom deltaker føler informasjonen gitt rundt testingen virker uklar, eller de sitter igjen med mer spørsmål som ikke er besvart gjennom introduksjonen, vil deltaker ha rett til å få oppklart videre detaljer de eventuelt har spørsmål om vedrørende testingsprosessen.

Deltaker har også rett til å endre eller få tilgang til sin egen data ved et hvilket som helst tidspunkt i prosessen.

Deltaker har når som helst mulighet for å trekke seg fra testingsprosessen dersom dette skulle være ønskelig. All eventuell innsamlet data ved aktuell deltaker vil deretter slettes.

Ved spørsmål eller ønske om å trekke seg kan kontaktperson kontaktes ved mailadresse, kristo9@stud.ntnu.no.

Dato: _____

Signatur: _____

Vedlegg B

Scenario og spørsmål.

Ved brukertestingen blir det tildelt et utvalg ulike scenarioer til aktuelle brukere. Disse scenarioene blir plukket ut slik at deltakeren gjennomfører to av disse under intervjuet. For å få et overordnet bilde av webapplikasjonen blir scenarioene til deltakere plukket ut tilfeldig slik at de går igjennom alle scenarioene ulikt i løpet av de tre intervjuperiodene.

Scenarioer.

1. Logg på kundebruker og finn leverandørene til vedkommende.
2. Logg på administratorbruker og endre bruker "Eksempel" til administrator.
3. Logg på ansattbruker og send mail til alle kundene med leverandører.
4. Logg på ansattbruker og finn leverandørinformasjon ved "Eksempelleverandør " til kunden "Onkel skrue".
5. Logg på ansattbruker og rediger stikkord ved "Onkel Skrue".
6. Logg på ansattbruker og naviger til svar på mailen med emne "Test123" ved kunde "Onkel Skrue".

Refleksjonsspørsmålene som stilles blir stilt ut fra hvordan deltakeren navigerer systemet og hvilke kommentarer deltakeren gir underveis i utførelsen. Det blir også stilt spørsmål om opplevelsen ved systemet, hvilke endringer de selv ville utført for et enklere oppsett, og hvilke tanker de sitter igjen med etter utførelsen av scenarioene.

Vedlegg C

Use case

C.1 High level use cases

Navn: Legge til ny kunde.
Aktør: Admin.
Mål: Lage en ny kunde, og fylle inn informasjon.
Beskrivelse: Trykker på “ny-kunde”-knapp. Den blir da tatt til en side der den kan legge inn informasjon om kunden. Informasjonen som må legges inn er navn og kontaktperson. Kan legges til kommentar, tags, leverandører og referanse.

Navn: Legge til ny leverandør.
Aktør: Admin.
Mål: Lage en ny leverandør, og fylle inn informasjon om den.
Beskrivelse: Admin trykker på “ny-leverandør”-knapp. Den blir da tatt til en side der den kan legge inn informasjon om leverandøren. Informasjonen som må legges til er leverandørnavn og kontaktperson. Kan legge til kommentar.

Navn: Se dashbord.
Aktør: Admin og ansatt.
Mål: Bruker har tilgang til dashbord ved applikasjonen.
Beskrivelse: Brukeren kommer inn på startsiden, der den ser aktuelle kunder listet opp. På siden kan den søke etter tags/navn og trykke på en kunde/leverandør for å se detaljer.

Navn: Redigere kunde.
Aktør: Admin og ansatt.
Mål: Redigere informasjonen ved en kunde.
Beskrivelse: Admin eller ansatt med ansvar for kunden går inn på en kundesiden og trykker på "rediger"-knappen. Den kan da endre informasjon ved de forskjellige informasjonsboksene. Når modifiseringen av informasjon er ferdig blir den sendt til databasen.

Navn: Legge til leverandør på kunden
Aktør: Admin og ansatt
Mål: Opprette en relasjon mellom kunden og leverandør
Beskrivelse: Bruker går inn på en kundeside og trykker på "legg til leverandør"-knappen. Den kan da velge en eksisterende leverandør og opprette en relasjon mellom kunden og leverandøren.

Navn: Redigere leverandør
Aktør: Admin og ansatt
Mål: Redigere informasjonen ved en leverandør.
Beskrivelse: Bruker går inn på en leverandørside og trykker på "rediger"-knappen. Den kan da endre dataen i de forskjellige informasjonsboksene. Når bruker er ferdig blir den oppdaterte informasjonen sendt til batabasen.

Navn: Lese mail til/fra kunde
Aktør: Admin og ansatt
Mål: Lese mailhistorikken ved en kunde
Beskrivelse: Bruker med tilgang til kunden kan se mailhistorikken ved kunden. Den skal også kunne se mailene til og fra de tilknyttede leverandørene.

Navn: Sende ny mail
Aktør: Admin og ansatt
Mål: Brukeren skal kunne sende mail
Beskrivelse: Bruker er på kundesiden til en kunde og trykker på "send mail"-knappen. Kan skrive inn emne, aktuelle mottakere, og tekst i informasjonsfeltene. Mailen sendes og lagres i databasen.

Navn: Send mail til flere
Aktør: Admin og ansatt
Mål: Bruker skal sende mail til flere kunder eller leverandører samtidig.
Beskrivelse: Bruker er på dashbord og huker av på kundene som skal mota mailen, før den trykker på "send mail". Bruker blir send til send-mail-siden og kundene er fylt inn i motager-feltet. Bruker kan også legge til eller fjerne motagere på mail-siden.

Navn: Autorisering / Innlogging
Aktør: Admin, ansatt og kunde
Mål: Aktøren logger seg på applikasjonen
Beskrivelse: Aktørene kommer til et startvindu der de kan logge seg på ved hjelp av Azure AD. Aktørene logger på og blir tatt til sin startside:

- Kunden sendes til sin kundesiden.
- Kunden blir sendt til dashbordet med oversikt over kunder.
- Admin blir sendt til en startside hvor den kan velge å se administatorrettigheter, eller dens kunder.

Navn: Se kundeside.
Aktør: Admin og ansatt.
Mål: Admin og ansatt skal kunne se kundesiden.
Beskrivelse: Aktør starter i dashbordet, den trykker på en kunde og blir tatt til kundesiden.

Navn: Se kundeside. (Kunde)
Aktør: Kunde.
Mål: Kunden skal se sin egen side med bare leserettigheter.
Beskrivelse: Kunden logger på og blir tatt til sin egen side. Den kan se informasjon om seg selv, men ikke gjøre endringer eller se mailfunksjoalitet.

Navn: Se leverandørside.
Aktør: Admin og ansatt.
Mål: Se leverandørsiden.
Beskrivelse: Aktør trykker på leverandør ved kunden for å komme til aktuell leverandørside.

Navn: Søke etter stikkord
Aktør: Admin og ansatt
Mål: Aktøren skal finne en kunder basert på stikkord
Beskrivelse: Aktøren er på dashbordet og bruker søk-funksjonen for å søke etter stikkord. Alle kunder med dette stikkordet vil komme opp.

Navn: Slette kunde
Aktør: Admin
Mål: Slette en valgt kunde
Beskrivelse: Aktøren navigerer seg til kundesiden og trykker på “slett”-knappen. All data om kunden vil bli slettet, inkludert kunde- og leverandørrelasjoner.

Navn: Slette Leverandør
Aktør: Admin
Mål: Slette en valgt leverandør.
Beskrivelse: Aktøren navigerer seg til leverandørsiden og trykker på "slett"-knappen. All data om leverandøren vil bli slettet, inkludert kunderelasjoner.

Navn: Svare på mail.
Aktør: Kunde og leverandør.
Mål: Svare på mail fra TietoEvry.
Beskrivelse: Mailen blir medsendt en lenke som kan trykkes på for å svare. Aktøren blir sendt til en svarside hvor den kan skrive inn og svare på mailen.

Navn: Respons på mail
Aktør: Kunde og leverandør
Mål: Sende en respons på en mottatt mail
Beskrivelse: Aktøren trykker på en link nederst i mailen den har mottatt. Den vil bli tatt til en side der den kan kommentere innholdet i mailen, eller bare trykke "sett".

Navn: Første pålogging
Aktør: Ansatt
Mål: Logger seg på applikasjonen og blir tildelt en rolle
Beskrivelse: En ansatt logger seg på systemet. Den ansatte får melding om å vente til en administrator har gitt den en rolle.

Navn: Gi rettigheter til ny ansatt
Aktør: Admin
Mål: Gi rettigheter til ny ansatt
Beskrivelse: Admin kan navigere ansatte ved ansattsiden og delegere rettigheter til ansatte.

C.2 Utvidet use case

Navn: Ansatte sender mail til en kunde.
Mål: En ansatt kan sende mail til en valgt kunde gjennom applikasjonens mail-funksjonalitet.
Pre-Betingelse: En ansatt er logget inn med internettilkobling og befinner seg på dashbordet for utførelse i samsvar med beskrevet programflyt nedenfor.
Post-Betingelse: Ansatt har navigert seg til mailfunksjonalitet og fylt ut informasjonsfelt før mail kan sendes.
Beskrivelse: Fra dashbordet kan den ansatte navigere seg til den aktuelle kunden for å svare på et eksisterende emne eller opprette en ny mail. Det er også mulig å opprette et nytt emne fra mailknappen på dashbordet.

---

1. Den ansatte benytter seg av mailfunksjonalitet på dashbordet og trykker på denne knappen.
    a. Aktør befinner seg ved dashbord.
    b. Aktør søker i kunden ved navn eller stikkord for å finne korrekt kunde. Kunder med tilsvarende stikkord eller navn vises vises.
    c. Aktør klikke på aktuell kunde og blir tatt til kundesiden.
    d. Aktør klikker på sendmail knapp ved kundesiden.
    e. Aktør legger til mottakere. Aktuell kunde er standard mottaker, men kan fjernes. Andre aktuelle mottakere er kunder og leverandører tilhørende aktøren som ønsker å sende mail.
    f. Aktør skrive emne og innhold.
    g. Aktør trykker på sendknapp.
    h. Mailen sendes til valgte mottakerne, og en kopi blir lagret i databasen.

Feil:
Ingen treff på navn eller stikkord.

1. Ansatt får ikke opp noen kunder som passer med søket.
2. Ansatt kan endre søket slik at det passer, eller lete manuelt gjennom listen.

Mailen mangler emne, innhold eller mottaker.

1. Ansatt trykker på send før alle informasjonsfelt er utfylt.
2. Det blir informert om manglende utfylt informasjonsfelt.
3. Ansatt går tilbake og fyller ut manglende informasjon før den kan sendes.

Vedlegg D

Kodestandarder

- Visual Studio Code programvareutvidelsen ”prettier” for formatering av koden.
- Enkel apostrof (’) istedenfor dobbel apostrof (”).
- Linjebredde på 120 tegn.
- Linjeinrykk på to blanke (tegn)
- Variable: drumedarCase
- Funksjonsnavn: drumedarCase
- React-funksjoner: PascalCase
- React-klassenavn: PascalCase
- If-er har krøllparentes uansett lengde.
- Alle funksjoner og klasser skal dokumenteres med JSDoc.
- *let* istedenfor *var* ved deklarering av variable
- Alle statements (setninger) avsluttes med semikolon, selv om det ikke er nødvendig.
- Variablenavn, kommentarer og dokumentasjon skrives på engelsk.
- Frontend
    - Bruke hovedsakelig vanlige funksjoner (ikke pilfunksjoner).
- Backend
    - Bruke pilfunksjoner
    - Bruke *module export* (JS) istedenfor *export* (TS) for å eksportere funksjoner.
        - Module export (JS) fordi Jest tester ikke støtter export (TS).

Vedlegg E

Prosjektskisse

Prosjektskisse

Bachelorgruppe 16

Alexander Aschlund Jørgensen
Øyvind Timian Dokk Husveg
Didrik Kielland Bjerk
Kristoffer Haugen

Institutt for datateknologi og informatikk
NTNU i Gjøvik
Norge
28.01.2021

Contents

1 Mål og Rammer

1.1 Bakgrunn

TietoEvry er et selskap som målsetter seg å bruke teknologi for å skape et digitalt fortrinn for virksomheter og samfunnet rundt oss. De forholder seg daglig til flere kunder og deres leverandører. Deres nåværende system medfører at oppbevaring av informasjon og kommunikasjon med de forskjellige leverandørene og kundene befinner seg på ulike plasser. Dette har blitt pekt ut som en tidskonsumerende og lite effektiv løsning, og gjennom denne bacheloroppgaven ønskes det å utarbeide en løsning for dette problemet. Ved å lage et system som samler relevant informasjon, og refererer videre til diverse annen informasjon, håper TietoEvry å effektivisere samhandlingen mellom konsulent/prosjektleder, kunde og leverandør. Systemet skal være tilgjengelig som en webapplikasjon.

Bacheloroppgaven beskrevet videre i denne rapporten omhandler utviklingen av et slikt system. Det vil bli utført drøftinger om forskjellige løsninger, begrunnelser ved utføringer, og dokumentasjon ved arbeid og videre planlegginger av prosjektet. Hvilke mål, rammer og omfang oppgaven innebærer, og hva som skal til for at denne oppgaven løses på en tilfredstillende måte.

1.2 Problemstilling

Systemet TietoEvry bruker for å kommunisere med kunder er ustrukturert og spredt. Informasjonskanalene er uavhengige av hverandre, og har i hovedsak ingen koblinger mellom hverandre. For å sikre god arbeidsflyt, bedre kommunikasjon mellom oppdragsigiver og kunder, og effektivisering av det nåværende systemet skal det nye systemet løse disse behovene. Det er behov for et nytt effektivisert system som gjør arbeidshverdagen lettere for både kundebehandlingen hos TietoEvry, og deres kunder.

1.3 Prosjektmål

Gjennom prosjektet settes konkrete og tydelige mål tidlig i planleggingsfasen. Dette gjøres for å ha mulighet til å jobbe målrettet mot konkrete mål gjennom hele prosjektet, og også teste resultatet opp mot disse målene i etterkant. Diverse mål deles inn i effektmål, resultatmål og læringsmål for prosessen.

1.3.1 Effektmål

Effektsmålene for oppgaven omhandler den graden effektiviseringen av det gamle systemet påvirker tid, ressurser, og penger ved de involverte partene.

- Redusere tiden det tar en ansatt å sende og dokumentere mail med cirka 1 time for hver kunde de overser per uke.
  - Etter diskusjon med oppdragsgiver ble det estimert at det er mulig å redusere tidsforbruket med cirka 1 time per kunde som oversees av

en ansatt i løpet av en uke.

- Redusere kostnader ved å effektivisere tid og mulighet for en mer effektiv dialog mellom TietoEvry og kunden. En mer oversiktlig dialog mellom oppgavetaker og kunde vil redusere muligheten for at viktig informasjon blir glemt eller oversett.
  - Redusering av kostnader kan måles opp mot tidsforbruk og antall feil i form av oversett informasjon underveis.
- Redusering av tid for opplæring, og dermed frigjøre ressurser i form av raskere opplæring.
  - Slik systemet er satt opp i dag er det veldig vanskelig å ha en oversikt over hvor ting befinner seg. Det krever læring og ressurser for å lære en ansatt systemet. Ved å redusere opplæringstid vil det frigi ressurser ved opplæring og samtidig få nyaansatte raskere opplært.
- Forbedre tilfredshet fra kundene sin side som et resultat av et mer oversiktlig og ryddig system, raskere svar og bedre kommunikasjon, raskere utførelse av arbeid.
  - Ved å benytte kundeundersøkelser kan det være mulig å dokumentere den graden systemet kan ha en effekt på kundetilfredshet.

1.3.2 Resultatmål

Målet for prosjektet er å utvikle en webapplikasjon som samler informasjon om TietoEvry sine kunder, og deres leverandører på et sted. Den skal også gjøre dialog mellom kunden og TietoEvry mer strukturert og lett tilgjengelig for begge parter.

1.3.3 Læringsmål

Læringmålene for prosjektet er definert i emnebeskrivelsen for bacheloroppgaven [4]. Læringsmålene gir en indikasjon på hvilke læremål som skal oppfylles i løpet av bacheloroppgaven.

Prosjektspesifikt

- Lære å anvende scrum-metodikken i en reel situasjon.
- Lære å planlegge og estimere tid i større prosjekter.
- Lære å samhandle med en oppdragsgiver.
- Lære å dokumentere og henvise til kilder.

Teknologi

- Lære å bruke tester og andre kontrollsystemer.
- Lære å anvende versjonskontroll, ved hjelp av git.

- Lære å anvende de forskjellige teknologiene, for eksempel skytjenester og databaser.
- Lære å designe brukervanlige og intuitive nettsider.

1.4 Rammer

Rammene for oppgaven defineres ved teknologiske og fysiske rammer. I tilegg blir eventuelle andre føringer definert for seg selv.

1.4.1 Fysiske rammer

- Perioden for gjennomførelse av bachelorprosjektet er preget av koronasituasjonen. Det blir stadig innført nye nasjonale tiltak, og disse kan påvirke gruppearbeidet i den grad det ikke lenger vil være mulig å møtes fysisk. Siden det ikke alltid er mulig å organisere fysiske oppmøter har kommunikasjonsplattformer som Microsoft Teams og Zoom blitt alternative plattformer for organiserte møter og diskosjoner.
- Prosjektperioden strekker seg fra 11.01.2021 til 20.05.2021. For å gjennomføre prosjektet slik det er ønsket vil det være viktig å konstant jobbe mot fristen, uten å havne for mye på etterskudd, gjennom hele perioden.

1.4.2 Teknologiske rammer

- Prosjektoppgaven utvikles slik at den kan benyttes av TietoEvry Brumunddal. Oppdragsgiver fokuserer ikke på at resultatet skal publiseres til andre enn deres egen virksomhet, selvom det burde være mulig å utvide applikasjonen på et senere stadiet slik at dette er mulig.
- Oppdragsgiver har informert om at oppgaven utføres etter eget ønske med tilnærmet frie rammer. Det er informert om anbefalte teknologier og verktøy for gjennomføring, men beslutningen om hvilke teknologier og verktøy som benyttes ligger hos oss. Se seksjon 2.2.
- Det er viktig at vi gjennom hele prosjektet forbeholder oss de overordnede lover og regler som er satt, i tillegg til de avtalte regler som er bestemt innad i gruppen og med andre involverte parter.
- Det er viktig at webapplikasjonen tilpasses de mest brukte nettleserene, både for mobil og desktop[1, 3, 8]. Tabbelen viser markedsandelen til forskjellige nettlesere, ekskludert mobile nettlesere, som tilsammen burde dekke 95% eller mer 1. Systemet skal fungere ved bruk av disse nettleserene.

1.4.3 Andre føringer

Andre føringer ved gruppeavtalen og de rammene avtalen reiser. Det kan bli nødvendig å definere nye rammer ved prosjektet på et senere tidspunkt, enten

Nettleser	Statcounter	W3Counter	NetMarketShare	Gjennomsnitt
Chrome	66,0%	65,3%	69,3%	66,9%
Safari	10,4%	16,7%	3,7%%	10,3%
Firefox	8,4%	4,4%	7,5%	6,8%
Edge/IE	9,4%	5,5%	13.3%	6,9%
Opera	2,6%	1,5%	1,2%	5,3%
Andre	3,2%	6,6%	5%	3,8%

Table 1: Markedsandelen til forskjellige nettlesere

av oppdragsgiver eller gruppen selv. Ved endringer skal dette dokumenteres.

2 Omfang

Bacheloroppgavens omfang er definert av oppgavebeskrivelsen, samt diskusjoner med oppdragsgiver. Det er nødvendig med en god forståelse av oppgavens krav og de utfordringene den medfører, for å sikre ønsket oppnåelse ved resultatet. Det er også viktig med en konsekvent dialog mellom gruppen, oppdragsgiver og veileder.

2.1 Fagområde

Fagområdet til applikasjonen går under prosjektstyring, kundehåndtering (oversikt) og kommunikasjon. Hovedoppgaven til applikasjonen er å organisere og strukturere data om kunder og leverandører. I tilleg til å bedre kommunikasjonen mellom kunder, leverandører og TietoEvry.

2.2 Avgrensning

Det er ønsket at det utvikles en minimumsløsning. Dersom det er tilstrekkelig med tid vil løsningen bli utvidet etter diskusjon med oppdragsgiver. Videre avgrenser oppgaven systemet til å utvikles som en webapplikasjon. Det er heller ikke mulighet for andre enn admin og utvalgte ansatte ved TietoEvry å legge til nye brukere i systemet.

2.3 Oppgavebeskrivelse

Den beskrevede oppgaven ønsker en webapplikasjon med formål om å samle dialog og relevant informasjon mellom kunder og TietoEvry gjennom et brukergrensesnitt som dirigerer brukeren til de diverse informasjonskaneler.

2.3.1 Innhold i applikasjonen

- Webapplikasjonen skal hovedsaklig fungere som en portal for konsulenter og prosjektledere for deres dialog med kunden. I tillegg skal det opprettes

en portal tilpasset kunden. Kundeportalen skal vise lagret informasjon om seg selv, samt kontaktinfo til leverandør og aktuell konsulent/prosjektleder ved TietoEvry, og mulighet til å sende mail. Konsulent/Prosjektleder vil ha mulighet til å vedlegge informasjon om kontaktperson, adresse, kontaktinformasjon og annen relevant informasjon ved opprettelse av brukerprofilen, eller ved en senere anledning. Senere skal kunden selv også ha mulighet til å oppdatere denne informasjonen om det skulle være nødvendig. Brukertilgang skal tilpasses bruker og videreformidle relevant informasjon til ulike brukere. Administrator vil ha tilgang til alt av informasjon, og vil i tillegg kunne administrere hvilke typer informasjonsfelt som lagres i databasen ved å legge til eller fjerne databasekolloner fra applikasjonen.

- Det skal være mulighet for å kommunisere gjennom webapplikasjonen, samt gi leserbekreftelse eller svar/tilbakemeldinger fra enten webapplikasjonen eller gjennom mail. Kommunikasjonen mellom kunden og konsulent/ prosjektleder skal vises i applikasjonen. Det er også spesifisert at mail skal kunne sendes ut til leverandører fra siden til en spesifikk kunde. Det skal være mulig å sende ut mail til en eller flere kunder og/eller leverandører samtidig.
- Det skal være mulighet for å legge ved stikkord i form av "Tags" til hver enkelt kunde. Ved å legge ved stikkord skal det være mulig å søke opp alle kunder med enkelte stikkord, og det skal være mulig å redigere ved å legge til eller fjerne stikkord ved kunden etterhvert.
- Brukergrensesnittet i applikasjonen skal være enkelt, ryddig, organisert, og tilpasset den aktuelle brukeren.

3 Prosjektorganisering

For å gjennomføre bachelorprosjektet etter de forventningene gruppen og oppdragsgiver har satt, vil det være viktig å utarbeide en tydelig prosjektorganisering fra første stund.

3.1 Ansvarsforhold og roller

Ansvarsforholdet ved prosjektarbeidet ligger hovedsaklig hos alle gruppemedlemmene. Krav om aktiv deltakelse, innspill, nødvendige forberedelser og en ryddig gjennomførelse etter gruppens enighet er blant de viktigste ansvarsforholdene hvert gruppemedlem til en hver tid skal følge.

Under prosjektutførelsen vil TietoEvry Brumundal være oppdragsgiver og Frode Haug ved NTNU vil fungere som veileder. Innad i prosjektgruppen vil alle gruppemedlemmer fungere som utviklere. Som utvikler har gruppemedlemmer ansvaret for å ferdigstille tildelte oppgaver i løpet av sprinten, men de skal også hjelpe resterende gruppemedlemmer om nødvendig. Det vil løpende bli rullert

på oppgaver som referatskriving, møteinnkallinger og oppgavedelegering etter enighet i gruppen.

Gjennom prosjektet vil Kristoffer Haugen være Scrum Master. Han har hovedansvaret for å kommunisere med product owner (oppdragsgiver) og veileder, på vegne av gruppa. Kristoffer Haugen vil ha ansvar for de ukentlige møtene og han vil ha en ekstra stemme ved valg som ender likt. Alexander Jørgensen vil være prosjektleder og sørge for fremdrift og passe på at gruppen når de delmålene som er satt til riktig tid.

3.2 Regler og rutiner i gruppen

Tanken ved opprettelsen av et reglement er å nedskrive forventinger og diverse krav gruppen stiller til hverandre for gjennomførelse av prosjektet. Dersom disse forventingene blir brutt vil gitte sanksjoner kunne utøves. Dette gjøres for å sikre en god harmoni og arbeidsmoral i gruppen gjennom hele prosjektet.

1. Det forventes at alle medlemmer i gruppen legger ned minimum 25 timer med arbeid hver uke.
2. Alle gruppemedlemmer skal loggføre tiden brukt på arbeidet i et verktøy bestemt av gruppen. Se seksjon 5.3.
3. Gruppemedlemmene skal være aktive i deltakelse ved arbeidet. Det vil si tilstedeværelse og konsentrasjon om oppgaven under avtalte møter. Det vil også medføre et ansvar for drive prosjektet videre.
4. Medlemmene skal møte til avtalte møter. Dersom et gruppemedlem ikke kan møte på avtalt tidspunkt skal det gis beskjed om dette så fort som mulig med begrunnelse.
5. Ved faglige uenigheter i gruppen vil flertallets stemme veie tyngst, og det må fattes lojalitet til det utfallet. Dersom utfallet blir likt vil Scrum Master sin stemme veie tyngst.
6. Gruppen plikter å være tidlig ute med informasjon ved misnøye hos enkelte gruppemedlemmer dersom man er misfornøyd med deres innsats gjennom prosjektet.

Dersom det skulle være slik at regelmessige brudd på reglene oppstår vil det være nødvendig å gjøre tiltak. Det er skrevet ned rutiner for brudd på regler innad i gruppen, og med veiledning fra veileder:

1. **Et brudd:** Møte med alle gruppedeltakere om problemet.
2. **To brudd:** Skriftlig advarsel fra de andre der:
    - Bruddet på reglement blir påpekt.
    - Konkret hva som kreves av arbeid for å rette opp bruddet.
    - Konkret om videre konsekvenser dersom det ikke bedres.

3. **Tre brudd:** Samtale med alle gruppemedlemmer og veileder.
4. **Tre/fire brudd:** Skriftlig ekskludering av gruppen i samvær med veileder.

4 Planlegging, oppfølging og rapportering

For å gjennomføre prosjektet på en ryddig og oversiktlig måte er det nødvendig å hele tiden ha en plan for gjennomførelsen. Ved å definere en tidlig plan for prosjektet passer vi på å ligge i rute for videre progresjon. Dersom en situasjon tilsier at en endring er nødvendig vil dette bli diskutert og dokumentert.

4.1 Hovedinndeling av prosjektet

Oppdragsgiver har kommet med et forslag til minimumsløsning av applikasjonen, i tillegg til en del foreslåtte utvidelser. Muligheter for utvidelser i løpet av prosjektperioden er noe av grunnen til valget av en smidig utviklingsmodell, som legger til rette for at kravspesifikasjonen kan endres underveis i utviklingsfasen. På grunn av koronapandemien vil muligheten til fysisk oppmøte redusereres. Det er derfor viktig for å organisere en klar plan fra start om hvordan arbeid fordeles, utføres og hvordan møtene blir organisert innad i gruppen og med oppdragsgiver/veileder. På grunn av dette, og fordi oppdragsgiver kom med et ønske om det, har det blitt vurdert at Scrum vil være en ideell utviklingsmodell for prosjektet.

Til å begynne med vil det legges stort fokus på å forstå hva oppdragsgiver ønsker fra applikasjonen, og få på plass en oversikt over alle "use casene". Videre vil vi presentere forslag til hvordan UI'en til applikasjonen skal se ut. Vi gjør dette såpass tidlig i prosjektet for å forsikre oss om at oppdragsgiveren og vi har samme forståelse om hvordan det endelige produktet skal bli. Brukerinteraksjon er en svært viktig del i applikasjonen, så tett samarbeid med oppdragsgiver blir avgjørende for at det endelige produktet skal oppfylle effektmålene og forventingene til oppdragsgiver.

Etter dette vil vi lage et diagram av databasen. Oppdragsgiveren har ytret et ønske om at det legges vekt på at dataen i databasen blir håndert på en skikkelig og strukturert måte. Oppdragsgiver har faglig kompetanse innen området, så de kommer til å være involvert i arbeid og endringer gjort i databasen.

I utviklingsfasen vil det legges fokus på å først utvikle en god minimumsløsning. De videre utvidelsene foreslått av oppdragsgiver lar seg dele opp i komponenter som kan implementeres uten store endringer i minimumsløsningen.

4.2 Plan for statusmøter og beslutningspunkter i perioden

Det er planlagt et møte med oppdragsgiver hver onsdag. Gjennom prosjektet kommer vi til å ha sprinter med varighet på en uke, og sprintmøtene blir

tatt sammen med oppdragsgiver. Sprintmøtene vil inkludere et møte med oppdragsgiver for å diskutere jobben som er gjort og avklare spørmål gruppedmdelemmene har om applikasjonen. Det er planlagt møte med veileder hver tirsdag 9:15-10. Veiledingen vil i all hovedsak handle om prosessen rundt arbeidet og prosjektrapporten. Hyppigheten på møtene kan reduserer utover i prosjektet hvis behovet for veiledning minker.

5 Organisering av kvalitetssikring

5.1 Dokumentasjon, standardbruk og kildekode

- All kode skal gjennomgås og bli kontrollert av minst et annet gruppemedlem.
- Kode, kommentarer og kodens dokumentasjon skal skrives på engelsk.
- Det skal brukes JSDoc til å dokumentere funksjoner og klasser [6].
- TypeScript standarder skal benyttes i stedet for JavaScipt der det er mulig [7].
- Det skal brukes en linter for å finne feil med koden før den commites. Se seksjon 5.3.

5.2 Testing

For testing av koden benyttes rammeverket for JavaScript testing "Jest" [2].

5.3 Verktøy

I tabell 2 er det en oversikt over verktøyene vi skal bruke.

Program/Verktøy	**Funksjon**
VSCode	IDE for utvikling av kode. Med Azure utvidelser for å utplassere applikasjonen og typescript utvidelser for feilsjekking
Jest	Testrammeverk for JavaScript
Linter	Gir bedre kodestandard/kvalitet og fører til mer effektiv koding
Bitbucket	Lagring og håndtering av kildekode og versjonskontroll
Azure Authentification	Verifisere at bruker er den de sier de er
Azure Active Directory	Lagrer informasjon om brukerne som blir brukt ved verifiseringen??
Azure SQL Database	Lagrer data
Azure Web Site	Brukergrensesnitt
Overleaf	Online skrive og redigeringsprogram for latex
Trello	For prosjektstyring og issuetracking
TeamGant	Online verktøy for å lage gantt-diagrammer
Clockify	For loggføring av arbeidstid

Table 2: Liste over verktøy

5.4 Risikoanalyse

Tabell 3 viser mulige risikoer under prosjektperioden. De har en sannsynlighetfaktor, og en konsekvensfaktor. De to faktorene vurderes utifra 1 til 3. Sannsynlighet visualiserer hvor sannsynlig hendelsen er for å inntreffe. Konsekvens viser graden av betydning utfallet har for prosjektet. Disse tallene blir multiplisert og denne totalen regnes som den totale risikoen ved en eventuell hendelse. Ansvarskollonen sier hvem som er ansvarlige for at risikoen skjer. B er bachelorgruppen, O er oppdragsgiver og I er ingen.

I tabell 4 er det laget tiltak for de hendelsene med høyest total-risiko. Tiltakene vil være med å redusere sannsynligheten eller konsekvensene av hendelsene.

Nr	Risiko	Ansvar	Sannsynlighet	Konsekvens	Total
1	Prosjektet blir ikke ferdig i tide	B	2	3	6
2	Feilestimering av nødvendig tid for gjennomførelse av oppgaver	B	3	1	3
3	Webapplikasjonen oppfyller ikke kravene fra oppdragsgiver	B, O	1	2	2
4	Tap av utarbeidet materiale som kode og rapport	B	2	3	6
5	Langvarig sykdom som hindrer deltakere i å jobbe med prosjektet	I	2	2	4
6	Konflikt blant gruppemedlemmene og/eller oppdragsgiver	B	1	3	3
7	Dårlig dokumentasjon og kvalitet på koden gjør applikasjonen vanskelig å vedlikeholde og videreutvikle	B	2	2	4
8	Oppdragsgiver endrer kravspesifikasjon under prosjektet	O	2	1	2
9	Utviklingsverktøy fungerer ikke som forventet	I	1	2	2

Table 3: Mulige risikoer

Nr	Beskrivelse	Tiltak
1	Ikke ferdig i tide	Om vi ser at vi ikke rekker å bli ferdig med alle de planlagte funksjonene til programmet, skal gruppen sammen med oppdragsgiver finne ut hvilke funksjoner som er viktigst å bli ferdig med. For å minske sannsynligheten for at vi ikke blir ferdige skal vi ha jevnlige møter som sprintplaning meetings der vi vurderer om prosjektet er forsinket.
2	Feilestimering av tid	Får å prøve å redusere feilestimeringen skal gruppen bruke planing poker ved estimering. Men siden de fleste av teknologiene vi skal bruke er nye for oss, er sannsynligheten for feilestimering fortsatt stor. For å forhindre at dette påvirker prosjektet i en stor grad vil vi ha ukentlige møter der utviklerne sier noe om hva de har jobbet med og om de har rukket å bli ferdig med alt. Om en oppgave tar mye lenger tid en først antatt vil flere på gruppen bistå for å få det ferdig.
3	Oppfyller ikke kravene	For å forsikre oss om at applikasjonen oppfyller kravene til oppdragsgiver skal det utvikles prosjektmål. Den skal inneholde effektmål og resultatmål som oppdragsgiver godkjenner, se seksjon 1.3.1 og 1.3.2. I tillegg skal vi ha ukentlige møter der de tester ny funksjonalitet for å sjekke om det virker som forventet.
4	Tap av materiale	For å unngå tap av kode vil koden bli lagret i BitBucket, lokalt hos alle utviklerne og eksternt i Microsoft sin Azure-cloud. Rapporten lagres i Overleaf [5] og det vil bli tatt back-up jevnlig; minst en gang i uka.
5	Sykdom	For å minske utfallet av at er person må forlate gruppen skal vi på de ukentlige møtene oppsummere hva hver av oss har gjort og kort forklare hvordan funksjonaliteten er implementert. Vi skal også unngå at en person jobber med en stor del av applikasjonen alen. For eksempel at bare en jobber med backend. Vi skal også være flinke til å dokumentere all kode, sånn at det blir lettere for andre å sette seg inn i de forskjellige delene av applikasjonen. Se punkt 7.
6	Konflikt	Om det oppstår en konflikt i gruppen, så skal man følge reglene som har blitt satt tidligere i prosjektet. Ser seksjon 3.2
7	Dårlig dokumentasjon	For å unngå problemer med uoversiktlig koden må all kode kommenteres og dokumenteres ved hjelp av JSDoc. Det skal også lages unit-tester for kritiske deler av systemet. Helst på så mye som mulig om en har tid. Både kommentering, dokumentering og kritiske tester må være implementert før koden kan sies done.

Table 4: Tiltak mot risikoene

6 Plan for gjennomføring

6.1 Gantt-diagram

Det første utkastet av Gantt-diagrammet er en tenkning om hvordan den ideele planen vil se ut. Det er satt opp sprinter på en uke. Grunnen til korte sprinter er hovedsaklig et ønske om hyppige ukentlige møter gjennom hele prosjektet, dette er forklart ved inndelingen av prosjektet i seksjon 4.1. Det er ønskelig at gruppen gjennom hele perioden jobber regelmessig slik at den tenkte prosessen passer best mulig.

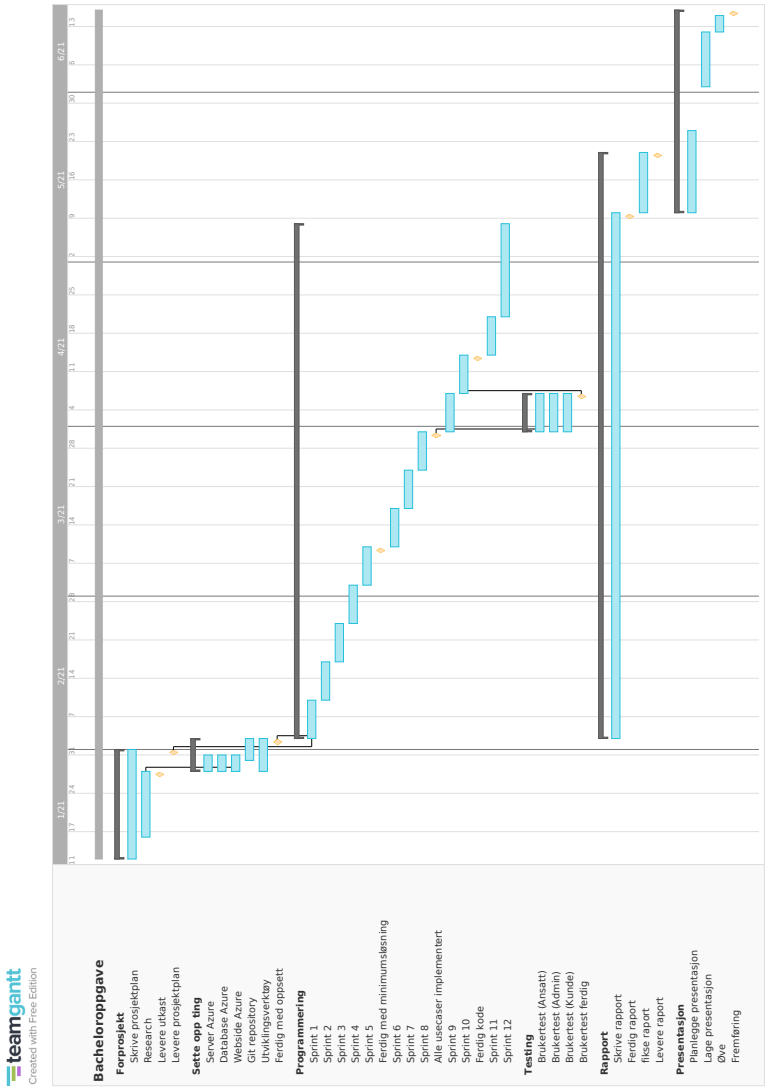


Figure 1: Plan for prosjektperioden

6.2 Milepæler og statusrapporter

- **Milepæl 1, 27.01.2021:** Utkast av prosjektskissen skal sendes til veileder og oppdragsgiver for tilbakemelding og evaluering.
- **Milepæl 2, 31.01.2021:** Endelig prosjektskissen skal leveres til veileder.
- **Milepæl 3, 02.02.2021:** Oppsett av alle teknologier og verktøy er ferdig.
- **Statusrapport 1, 17.02.21**
- **Milepæl 4, 09.03.2021:** Minimumsløsningen er implementert.
- **Statusrapport 2, 24.03.21**
- **Milepæl 5, 30.03.2021:** Alle use caser er implementert.
- **Milepæl 6, 7.04.2021:** Brukertester skal være fullført.
- **Milepæl 7, 13.04.2021:** Koden skal være ferdig, kun bugfiksing og andre små endringer gjenstår.
- **Statusrapport 3, 28.04.21**
- **Milepæl 8, 9.05.2021:** Rapport skal være ferdig, bare småendringer gjenstår.
- **Milepæl 9.05.2021:** Rapport skal være ferdig, bare småendringer gjenstår
- **Milepæl 10, 20.05.2021:** Levere rapport
- **Milepæl 11, xx.06.2021:** Presentere bacheloroppgaven.

References

[1] Statcounter. Browser marketsandel. Hentet 20/01/2021.

[2] JEST. Hentet 26/01/2021.

[3] Netmarketshare. Browser marketsandel. Hentet 20/01/2021.

[4] NTNU. Emnebeskrivelse av bidat. Hentet 29/01/2021.

[5] Overleaf. Overleaf latex rapport. Hentet 17/01/2021.

[6] TypeScript. *JSDoc Reference*. Hentet 17/01/2021.

[7] Typescript. Typescript documentation. Hentet 25/01/2021.

[8] W3Counter. Browser marketsandel. Hentet 20/01/2021.

Vedlegg F

Prosjektavtale

NTNU

Norges teknisk-naturvitenskapelige universitet

Vår dato Vår referanse

Prosjektavtale

mellom NTNU Fakultet for informasjonsteknologi og elektroteknikk (IE) på Gjøvik (utdanningsinstitusjon), og

Jan Fredrik Gundersen, Guro Storlien Evensen

TietoEVRY (oppdragsgiver), og

Diderik K. Bjørk Kristoffer Haugen

Alexander A. Jørgensen

Øyvind Timian ~~Husveg~~ Dokk Husveg (student(er))

Avtalen angir avtalepartenes plikter vedrørende gjennomføring av prosjektet og rettigheter til anvendelse av de resultater som prosjektet frembringer:

1. Studenten(e) skal gjennomføre prosjektet i perioden fra 11.01.21 til 15.06.21.

   Studentene skal i denne perioden følge en oppsatt fremdriftsplan der NTNU IE på Gjøvik yter veiledning. Oppdragsgiver yter avtalt prosjektbistand til fastsatte tider. Oppdragsgiver stiller til rådighet kunnskap og materiale som er nødvendig for å få gjennomført prosjektet. Det forutsettes at de gitte problemstillinger det arbeides med er aktuelle og på et nivå tilpasset studentenes faglige kunnskaper. Oppdragsgiver plikter på forespørsel fra NTNU å gi en vurdering av prosjektet vederlagsfritt.

2. Kostnadene ved gjennomføringen av prosjektet dekkes på følgende måte:
   - Oppdragsgiver dekker selv gjennomføring av prosjektet når det gjelder f.eks. materiell, telefon, reiser og nødvendig overnatting på steder langt fra NTNU i Gjøvik. Studentene dekker utgifter for ferdigstillelse av prosjektmateriell.
   - Eiendomsretten til eventuell prototyp tilfaller den som har betalt komponenter og materiell mv. som er brukt til prototypen. Dersom det er nødvendig med større og/eller spesielle investeringer for å få gjennomført prosjektet, må det gjøres en egen avtale mellom partene om eventuell kostnadsfordeling og eiendomsrett.

3. NTNU IE på Gjøvik står ikke som garantist for at det oppdragsgiver har bestilt fungerer etter hensikten, ei heller at prosjektet blir fullført. Prosjektet må anses som en eksamensrelatert oppgave som blir bedømt av intern og ekstern sensor. Likevel er det en forpliktelse for utøverne av prosjektet å fullføre dette til avtalte spesifikasjoner, funksjonsnivå og tider.

4. Alle bestått bacheloroppgaver som ikke er klausulert og hvor forfatteren(e) har gitt sitt samtykke til publisering, kan gjøres tilgjengelig via NTNUs institusjonelle arkiv NTNU Open.

Tilgjengeliggjøring i det åpne arkivet forutsetter avtale om delvis overdragelse av opphavsrett, se «avtale om publisering» (jfr Lov om opphavsrett). Oppdragsgiver og veileder godtar slik offentliggjøring når de signerer denne prosjektavtalen, og må evt. gi skriftlig melding til studenter og instituttleder/fagenhetsleder om de i løpet av prosjektet endrer syn på slik offentliggjøring.

Den totale besvarelsen med tegninger, modeller og apparatur så vel som programlisting, kildekode mv. som inngår som del av eller vedlegg til besvarelsen, kan vederlagsfritt benyttes til undervisnings- og forskningsformål. Besvarelsen, eller vedlegg til den, må ikke nyttes av NTNU til andre formål, og ikke overlates til utenforstående uten etter avtale med de øvrige parter i denne avtalen. Dette gjelder også firmaer hvor ansatte ved NTNU og/eller studenter har interesser.

5. Besvarelsens spesifikasjoner og resultat kan anvendes i oppdragsgivers egen virksomhet. Gjør studenten(e) i sin besvarelse, eller under arbeidet med den, en patentbar oppfinnelse, gjelder i forholdet mellom oppdragsgiver og student(er) bestemmelsene i Lov om retten til oppfinnelser av 17. april 1970, §§ 4-10.

6. Ut over den offentliggjøring som er nevnt i punkt 4 har studenten(e) ikke rett til å publisere sin besvarelse, det være seg helt eller delvis eller som del i annet arbeide, uten samtykke fra oppdragsgiver. Tilsvarende samtykke må foreligge i forholdet mellom student(er) og faglærer/veileder for det materialet som faglærer/veileder stiller til disposisjon.

7. Studenten(e) leverer oppgavebesvarelsen med vedlegg (pdf) i NTNUs elektroniske eksamenssystem. I tillegg leveres ett eksemplar til oppdragsgiver.

8. Denne avtalen utferdiges med ett eksemplar til hver av partene. På vegne av NTNU, IE er det instituttleder/faggruppeleder som godkjenner avtalen.

9. I det enkelte tilfelle kan det inngås egen avtale mellom oppdragsgiver, student(er) og NTNU som regulerer nærmere forhold vedrørende bl.a. eiendomsrett, videre bruk, konfidensialitet, kostnadsdekning og økonomisk utnyttelse av resultatene. Dersom oppdragsgiver og student(er) ønsker en videre eller ny avtale med oppdragsgiver, skjer dette uten NTNU som partner.

10. Når NTNU også opptrer som oppdragsgiver, trer NTNU inn i kontrakten både som utdanningsinstitusjon og som oppdragsgiver.

11. Eventuell uenighet vedrørende forståelse av denne avtale løses ved forhandlinger avtalepartene imellom. Dersom det ikke oppnås enighet, er partene enige om at tvisten løses av voldgift, etter bestemmelsene i tvistemålsloven av 13.8.1915 nr. 6, kapittel 32.

12. Deltakende personer ved prosjektgjennomføringen:

NTNUs veileder (navn): Frode Haug

Oppdragsgivers kontaktperson (navn): Jan Fredrik Gundersen
Guro Storien Evensen

Student(er) (signatur): Didrik R. Bjørk dato 12/01-21

Alexander A. Jørgensen dato 12/01-21

Øyvind Timian Dokk Husveg dato 12/01-21

Kristoffer Haugen dato 12/01-21

Oppdragsgiver (signatur): Jan Fredrik C dato 12/01-21

Signert avtale leveres digitalt i Blackboard, rom for bacheloroppgaven.
Godkjennes digitalt av instituttleder/faggruppeleder.

Om papirversjon med signatur er ønskelig, må papirversjon leveres til instituttet i tillegg.
Plass for evt sign:

Instituttleder/faggruppeleder (signatur): _____ dato _____

Vedlegg G

Sprint Log

Sprint: Forprosjekt
Dato: 12.01 - 31.01
Sprintfokus: Prosjektplanlegging. Oppstartsfase.
Beskrivelse: Gruppenmedlemmene leste seg opp på de teknologiene som skulle benyttes gjennom prosjektet. Gruppen satte en møteplan for oppdragsgiver og veileder, og hadde samtaler med oppdragsgiver og veileder, gjorde valg angående hvilke teknologier og verktøy som skulle benyttes i løpet av prosjektet. For eksempel hvilken utviklingsmodell og rammeverk. Ferdigstilte prosjektplanen.

Sprint: 1
Dato: 03.02 - 10.02
Sprintfokus: Grunnleggende funksjonalitet.
Beskrivelse: Utarbeidet use case. La til user stories inn i backlog for fremtidige sprinter og arbeidsprosess. Satte opp enkel nettside med autentisering. Database og server er satt opp. Opprettet react-applikasjon.
Evaluering: Ble ferdig med det meste, rimelig fornøyde med innsatsen.

Sprint: 2
Dato: 10.02 - 17.02
Sprintfokus: Rapportskrivning og systemdesign.
Beskrivelse: Lagde resten av use casene og alle på gruppen leste gjennom use casene for å forsikre seg om at alle var på samme side. Skriving av sikkerhetskrav, omfang og operasjonelle krav. Databasemodellering og generell design av nettsiden ble ferdigstilt.
Evaluering: Mange på gruppen følte at user storiene som ble gitt i denne sprinten var litt vage og dermed at til neste sprint skal det være mer konkrete og spesifikke mål for sprinten. I tillegg kom gruppen fram til at det alle på gruppen skal møtes hver mandag og fredag kl 09.00, i tillegg til allerede bestemt møte ved TietoEvry Brumunddalen hver onsdag.

Sprint: 3
Dato: 17.02 - 24.02
Sprintfokus: Kobling av komponenter, design
Beskrivelse: Kobling mellom server og database og det ble gjort en del research på hvordan man skal forhindre at uønskede brukere får tilgang til backend og så implementere det. Lagt inn use caser i backlog, i tillegg jobbet gruppen med å hente informasjon og legge inn informasjon i databasen via funksjoner i serveren. Det ble også ferdigstilt design og implementasjon av dashboard og startsiden. Et av målene for sprinten var også å vise informasjon fra databasen i dashboardet. Dette ble ikke helt ferdig i tide på grunn av en bug.
Evaluering: Alle gruppemedlemmene var fornøyde med sprinten. Nesten alt ble ferdig, det eneste som sto igjen var nesten ferdig implementert. Gruppen diskuterte forbedringspotensiale kom med noen forslag:

- Komme til riktig tid
- Ha med mat (mange dro tidlig på grunn av at de ikke hadde med mat)
- Bestille rom på campus tidlig
- Registrere tid i Clockify

Sprint: 4
Dato: 24.02 - 03.03
Sprintfokus: Opprydding, vedlikehold, nye backendfunksjoner
Beskrivelse: Ferdigstilte forslag til navn til applikasjonen, ryddet litt i backloggen, la til beskrivelse på mange use caser, fikk godkjent navn av oppgavegiver. Design av logo og kundeside ble ferdigstilt. I frontend ble det gjort en del kodevedlikehold, som å legge til kommentarer hvor det var nødvendig. Funksjoner for å endre rettigheter på bruker, søk i databasen, legge til kunde og hente kundeinformasjon ble lagt til i API.
Evaluering: Mange av oppgavene i sprinten ble ikke ferdig, men gruppen følte fortsatt at det hadde vært en god arbeidsinnsats. Gruppen hadde et seminar i et annet fag, som gjorde at ingen kunne på jobbe på mandagen, det ble ikke tatt med i estimeringen. Dermed kom gruppen fram til at det var feil med estimering av hvor mye som kunne bli gjort denne sprinten.

Sprint: 5
Dato: 03.03-10.03
Sprintfokus: Nye backendfunksjoner, research på forskjellige emner
Beskrivelse: I denne sprinten så ble backend funksjonene dokumentert og en del research på mail ble gjort. I tillegg så ble alle gruppemedlemmene ferdig med å lese gjennom en hel bachelorrapport. Det ble gjort litt research og forsøk på å implementere kode tester for både frontend og backend. Det ble implementert flere frontend funksjoner som kaller API-et. Gruppen definerte også en kodestandard for prosjektet. Det ble og lagt til funksjoner for å hente informasjon kunder og dannelse av nye brukere.
Evaluering: Gruppen følte at arbeidet is sprinten var ok, men ifølge Gantt-diagrammet laget i starten av prosjektet, så skulle det vært ferdigstilt en prototype av applukasjonen i løpet av dagen. Det var den ikke, så det hadde blitt feilestimert hvor mye arbeid som ville være ferdig for å ha en prototype på dette tidspunktet, men det ble bestemt at innen neste uke så burde det være mulig å ha ferdigstilt en prototype.

Sprint: 6
Dato: 10.03 - 17.03
Sprintfokus: Mail funksjonalitet, roller og redigering av kunder
Beskrivelse: Her ble det laget design for en side der man kan registrere og redigere kunder og en side for å legge til nye leverandører. Disse sidene ble også implementert, i tillegg ble det lagt til funksjonalitet for mail systemet (motta og sende mail) i backend og de nye sidene. Redirect etter login ble også fikset slik at man blir sendt til forskjellige sider etter hvilken rolle man har.
Evaluering: Gruppen følte at det hadde vært middels innsats i løpet av sprinten. Ikke alt som skulle vært gjort ble ferdig, mye grunnet at ting tok lenger tid enn forventet og/eller vi var uforberedt på problemene.

Sprint: 7
Dato: 17.03 - 24.03
Sprintfokus: API funksjonalitet
Beskrivelse: Det ble lagt til en del ny API funksjonalitet, blant annet sletting av leverandør, kunde eller og ansatte. Mangler mer info?????
Evaluering: De fleste oppgavene for sprinten ble gjennomført, noen av dem trenger litt ekstra finpuss.

Sprint: 8
Dato: 24.03 - 07.04
Sprintfokus: Påskeferie, rapportskrivning, dokumentasjon, SSL sertifikasjon
Beskrivelse: Dette ble en to uker lang sprint på grunn av at påskeferien. Gruppen valgte å ta påskeferie, men å jobbe med rapporten i løpet av ferien.
Det ble satt opp API dokumentasjon. I løpet av sprinten hadde vi et møte med oppdragsgiver. Møtet handlet for det meste om et par spørsmål bachelorgruppen hadde rundt et par uklarheter, i tillegg kom oppdragsgiver med et par tips rundt SSL sertifikasjon. Resultatet av møtet var at bachelorgruppen og oppdragsgiver er på samme bølgelengde. I løpet av ferien skulle hvert gruppemedlem komme godt i gang med et kapittel av rapporten. Kapitlene ble fordelt etter preferanse og hvem som hadde jobbest mest med temaene.
Evaluering: Gruppen følte at det meste hadde blitt gjort, det var en vellykket sprint og gruppen følte seg i god form etter påskeferien.

Sprint: 9
Dato: 07.04 - 14.04
Sprintfokus: Fortsette på rapport, og videreutvikle nettsiden.
Beskrivelse: Prøve å implementere interasjonalisering, litt forbedringer på nettsidens funksjonalitet som søkebar og feilmeldinger, SSL sertifikat ettersom det ikke ble gjort forrige sprint. I tillegg er det lagt opp for skriving på rapporten.
Den planlagte arbeidsmengden ved sprinten er noe redusert grunnet sykdom ved gruppen. For å ikke overbelaste arbeidsmengde og tilpasse situasjonen har gruppen valgt å fokusere videre på nettsiden.
Evaluering: God innsats tross sykdom hos flere gruppemedlemmer. Sykdom har gjort det vanskeligere å møtes, både fysisk og over nett. Gruppen fikk gjennomført det aller meste som ble planlagt ved forrige sprint, men tross situasjonen er gruppen fornøyd med utførelsen av sprinten.

Sprint: 10
Dato: 14.04 - 21.04
Sprintfokus: Skrive videre på rapporten og oppdatere databasen
Beskrivelse: Videre skriving på rapporten, implementere funksjonalitet som lar admin endre på databasen gjennom applikasjonen.
Evaluering: Fikk ikke gjennomført alt slik vi ønsket, men hadde også større deloppgaver som omhandlet å skrive ferdig kapittel til rapporten.

Sprint: 11
Dato: 21.04 - 28.04
Sprintfokus: Skrive ferdig utkast til rapporten og starte redesign av siden..
Beskrivelse: Til mandag skal vi levere utkast til rapport til Frode. Går igjennom alt og skriver kapitlene "Implementasjon"og Deployment". I tillegg begynner frontend utviklerne å jobbe på å redesigne nettsiden, sånn at det skal være lettere/intuitivt å bruke nettsiden. I tillegg skal redesignet av nettsiden føre til at nettsiden blir responsiv.
Evaluering: Brukt mye tid på rapportskriving de siste dagene, pga innlevering av utkast til Frode. Gruppen jobbet godt med oppgavene som ble satt opp, men noen av disse, som kapitlene og revideringen av design og responsivitet tok mye lengre tid enn antatt. Mye jobb, men gruppen evaluerer sprinten til å være godt gjennomført.

Sprint: 12
Dato: 28.04 - 05.05
Sprintfokus: Forskjellige systemtester, rapportskriving, revidering av redesign av siden
Beskrivelse: Håper å komme godt igang med systemtester (unit tester, integrasjon tester og brukertester). Gjennomføre revideringen av design og responsivt slik som forrige gang. Gjøre endringer ved rapporten etter tilbakemelding fra Frode.
Evaluering: Fikk gjennomført de største aspektene ved sprinten. Brukte mer tid på rapport enn først antatt, og hadde to evalueringsmøter med veileder. Fikk gjennomført det viktigste, men har lagt til mye ved sprinten. Underestimert rapportskriving, og design av siden. Alt i alt, godt arbeid og mye ble gjort.

Sprint: 13
Dato: 05.05 - 12.05
Sprintfokus: Rette på feil og småjusteringer etter brukertest. Rapportskriving. Siste design av siden.
Beskrivelse: Ettersom unit testing av frontend ikke ble implementert forrige uke, skal dette sees på. I tillegg skal backend unit og integrasjon testene ble revidert for å få mer komplette tester. Videre skal det også skrives på rapporten, og vi satser på at nettsiden skal være omtrent ferdig.
Evaluering: Fått gjennomført de fleste oppgavene som var satt opp. Har en uke på finpuss av rapporten, og alt fokuset vil bli rettet mot rapportskriving. Alt i alt, en godt gjennomført sprint og alle arbeidet godt gjennom uken.

Sprint: 14
Dato: 12.05 - 19.05
Sprintfokus: Skrive på rapporten og fjerne unødvendige kodesnutter fra koden i både frontend og backend.
Beskrivelse: Målet ved denne sprinten er å bli ferdig med rapporten, og om gruppen føler at rapporten er godt gjort og det er mer tid til overs, så vil det jobbes litt på nettsiden.
Evaluering:

Vedlegg H

Møtereferater

Møte 12.01

Hvem: Oppgavegiver

Sted: Tieto Evry Brumunddal

Agenda: Agendaen for møtet dreier seg i hovedsak om å drøfte den gitte bacheloroppgaven i plenum med oppgavegiver. Skaffe en oversikt og ha mulighet til å kunne stille de spørsmålene som måtte oppstå for å sikre at oppgavegiver og oss som oppgavetakere er samordnet i tankeprosessen rundt oppgaven.

Hovedpunkter: Hovedpunktene for møtet dreide seg i hovedsak om å høre oppgavegiver sine tanker om hvordan oppgaven skal gjennomføres og hva de forventer gjennom bachelorperioden. Fikk noen tips om oppstart og hvilke verktøy som kan være aktuelle å benytte i utviklingen av prosjektet.

- Lagde raskt Use Case og diskuterte forskjellige roller i tjenesten. Admin, Reader, Kundeadmin, Kunde.
- Se videre på Use Case senarioer og skissere bunnlaget i oppgaven grundig før arbeidet begynner. Benytte Microsoft Azure for autentifisering og skydatabase, eventuelt Auth0.
- Se litt på hvilke verktøy og språk det er mulighet for å bruke. Ble nevnt PhP, Jira, NodeJs, Java, og mange flere ut ifra det vi selv måtte ønske.
- Har mulighet til å utvide prosjektet. Må estimere hvor mye tid utvidelsen vil ta.

Avtalt møte/arbeidsdag hos TietoEvry Brumunddal de neste onsdagene.

Videre arbeid:

- Bestemme oss for utvidelse av det opprinnelige prosjektet.
- Se dypere på hvilke teknologier/språk det er ønskelig å ta i bruk.
- Skrive prosjektavtale.

Møte 13.01

Hvem: Veileder, Frode Haug

Sted: Zoom

Agenda: Agendaen for møtet dreier seg om å starte opp, stille spørsmål og få tips for å få en god start på bachelorperioden. Drøfte noe om omfanget av den gitte oppgaven, og høre veileder sine synspunkt om mulighetene for å utvide den originale oppgaven.

Hovedpunkter:

- Tips: Treffes mye uavhengig av koronasituasjonen.
- Oppgaven burde utvides noe ut ifra det som allerede er tildelt av oppgaver. Veileder mener den ikke nødvendigvis er stor nok for å nå en A.
- Lage grupperegler med konsekvenser for brudd av disse. For å ha gode og tydelige retningslinjer for hva vi i gruppen forventer av hverandre.
- Deler av prosjektplanen kan benyttes i første kapittel av rapporten dersom det blir gjort bra med en gang. Dette vil spare en del tid.
- Kapittel to av rapporten omhandler kravspesifikasjon. Dersom vi benytter oss av Scrum og sprinter vil backlogen ta stor del av dette kapittelet. Kravspek kan bli skrevet tidlig i prosjektet.
- Skal skrives statusrapporter ca. 1.feb, 15.mars, og 1.mai. Inneholder bare hvordan vi mener vi ligger an.

Har satt av møte hver tirsdag klokka 09.15 med veileder. Vi vil lede møtene med de spørsmålene vi måtte ha. Veileder er der for å besvare spørsmål om prosessen, rapportskriving og diverse ting vi måtte lure på. Ikke nødvendigvis spørsmål direkte rettet mot WebTeknologi.

Videre arbeid:

- Levere prosjektavtale på BlackBoard.
- Jobbe med prosjektplanen. Snakk om cirka 20 sider. Når denne er ferdig sendes den til Frode.
- Delegere arbeidsoppgaver og forberede oss til å begynne på selve prosjektet. Ved siden av prosjektplanen er det lurt å se nærmere på arbeidsmetoder og språk, samt få en god oversikt over disse.
- Hvert gruppemedlem oppretter et Excel-dokument hvor deres arbeidsmengde blir dokumentert gjennom hele prosessen. Hvor mange timer ble det arbeidet, og hva ble gjort?

Møte 14.01, Lynkurs 1

Hvem: Bacheloransvarlig, Tom Røise

Sted: Teams

Agenda:

Se på karakteristika rundt oppgaven. Diskutere prosjektplan og hvordan det utarbeides. Hvordan velge systemutviklingsmodell. Diverse råd og vink for å utføre oppgaven og komme best mulig i gang.

Mål for møte: Gi noen holdepunkter for å komme i gang med arbeidet på en systematisk og strukturert måte. Prosjektplanen som skal utarbeides til 31.jan.

Hovedpunkter:

Generelle notater.

- Selvvalgte, små og homogene grupper med uerfarne deltakere.
- Cirka tid beregnet for arbeid. 3 * 30timer * 18 uker = +/-1600timer
- Må mestre å forholde oss til emnebeskrivelsen, retningslinjer, avtaler, veileder, oppdragsgiver og hverandre.
- Karaktervurdering legges ut. Sensur av ingeniørfag. Les igjennom, og være klar over punktene gjennom hele prosjektet.
- Dokumenter og vurderinger som gjøres underveis skal legges inn i rapporten.
- Et godt produkt er en forutsetning for en god rapport.

Planlegging.

- Avklare hva som er målet for prosjektet.
- Problemstilling, Gruppas målsetting angående oppfyllelse av læringsutbytte.
- Få et redskap til å styre mot målet.
- Disponere resursene fornuftig. Redusere risiko, Ha et godt beslutningsgrunnlag under prosjektdelen.
- Aktuell mal ligger ute.
- Rollene burde fordeles slik at alle får bidratt på hele prosjektet.
- Hva skjer om noen blir sjuk, er borte, ikke innfrir? Regler samt rutiner.
- Finne ut detaljert metode å benytte Scrum. Hvorfor? Hvordan? Hvem er med?
- Kvalitetssikring og regler—> Hold det håndterlig.

Valg av utviklingsmodell

- Se på alle, og deretter velg modell ut fra prosjektet.
- Se på: Universitetets krav til prosjekt. Karakteristika til oppgaven. Motivasjon og ferdigheter til deltakere. Ønsker og krav fra oppdragsgiver.
- Velg modell som passer til disse punktene. Hvorfor og hvordan? Prosjektplan del 4.

Råd for prosjektet.

- Lag en prosjektplan for gruppa. Ikke for veileder/oppdragsgiver. Ikke nødvendigvis "teoretisk korrekt".
- Møt hverandre fysisk.
- Ha en prosjektleder.
- Diskuter forventinger, ambisjonsnivå nå.
- Benytt oppdragsgiver og veileder.
- Fokuser på det kritiske.
- Før timelogg fra dag 1, korte referater og dokumenter valg.
- Les rapporter fra gode beslektede prosjekter tidlig!
- Start med oppgaver og ikke bruk all tid på prosjektplanen.
- Være nøye med kildebruk! Har blitt mer nøye med senere rapporter.

Videre arbeid:

- Utarbeide en prosjektplan innen 31.jan.
- Utarbeide timelogg og dokumenter alle valg underveis.
- Finne gode beslektede rapporter tidlig i prosessen. Spør veileder om dette før møte neste uke? Står også noen gode i presentasjonen fra møtet.

Møte 19.01

Hvem: Veileder, Frode Haug

Sted: Teams

Agenda: Snakke fremgangen i prosjektet. Stille spørsmål om prosjektskisse.

Hovedpunkter:

Lage to Gantt-diagrammer. Den første hvordan vi tenker nå, og senere den som faktisk ble. Deretter sammenlikne disse to i rapporten.

Frode forklarte litt rundt punktene fra lynkurs 1:

- 1.2 Prosjektmål
    - Effektmål: Hva er det kunden tjener på applikasjon.
    - Resultatmål: Hva er det systemet vårt er, og hva skal den gjøre for Evry.
    - Læringsmål: Hva skal vi lære/hva får vi lære? *Samarbeide i gruppen/med arbeidsgiver.*
- 1.3 Rammer
    - Praktiske rammer. Hvordan er rammene med tanke på Covid19? Hvordan jobber vi? Hvordan ligger ytre rammer til rette for arbeidet? Plattformer, teknologi, etc. arbeidsgiver krever at vi bruker.
- Ca. 2 A4 ark på Omfang: 2.1 2.2 2.3 i lynkyrs1. Gjør det grundig! Mye kan brukes i iden endelige rapporten hvis det blir bra.
- 2.1 Fagområde

    - Hvor innen dataverdenen er det oppgaven vår ligger. Kommunikasjon, web.. Sett scenen.
- 2.2 Avgrensing
    - Hva skal vi jobbe med innenfor fagområdet.
- 2.3 Oppgavebeskrivelse:
    - Kan hente ting rett fra oppgavebeskrivelsen fra Evry
- 4
    - Valg av SU-modell. *Eks: scrum, osv..*
- 5.2
    - Git, etc.

Videre arbeid:

- Fortsette jobb med research.
- Sende inn et utkast på prosjektskisse innen mandag 25.01.

Møte 20.01

Hvem: Oppgavegiver

Sted: TietoEvry Brumunddal

Agenda:

Forhøre oss om valgte teknologier for prosjektet. Høres det bra ut? Velge noe som oppgavegiver har kjennskap til (Med tanke på veiledning videre i prosjektet)? Utdypelse av prosjektet. Høre om de forskjellige igjen.

Hovedpunkter:

- Få tilgang til Azurekonto og nettsiden.
- Sette opp en enkel nettside og db.
- Se videre på Use Case, model view controller (MVC)
- Tegne opp modeller av database.
- Se på enkel UI?

Møte 27.01

Hvem: Oppgavegiver

Sted: TietoEvry Brumunddal

Agenda:

Sett på hvordan vi kan sette opp Azure server/functions. Fått tilbakemelding på designav webside og på målene og risikoanalysen i prosjektskissen.

Hovedpunkter:

Koden:

- Å lage en funksjon der man kan sende mail til alle kunder en leverandør har er ikke nødvendig, men det er bra
- Det kan være lurt å dele opp funksjonene i read og write (mer research nødvendig)

Rapporten:

- Effektmål:
    - Kan spare rundt 3timer i uka per person med å bruke det nye systemet. (1250kr/t)
    - Kan gi økt kundetilfredshet, pga bedre flyt / kommunikasjon. Som igjen kan føre til mersalg og dermed tjene mer.
    - Gjennom det nye systemet vil det bli bedre oversikt. Dette kan være med på å avdekke feilfakturering. Bedre kontroll og oppfølging av hva som er solgt.
- Resultatmål:
    - Hvem er aktuelle brukere.
    - Forklare alle ord i utvidet form (for eksempel ikke skrive «bruker» men forklare hvem de er).
    - Dele opp målene, skrive fra flere synsvinkler. (Dette blir lettere for kunden, dette blir lettere for oppdragsgiver).
- Risikoanalyse
    - Hvem er ansvarlige for risikoene. (for eksempel oss, oppdragsgiveren)
    - Tap av progresjon bør omformuleres, for eksempel «tap av utarbeidet materiale».
    - Sett av tid til retroperspektiv (hva har vi gjort bra, hva kunne vært bedre). Legge dette inn i sprint planning meeting, kanskje skrive noe om det i rapporten?
    - Punkt 7, bruk ordet kvalitet
    - Legge til noe om tekniske utfordringer, spør Frode.

Div:

Navn-ideer:

- Workcom.com.com
- Komder
- Kommunikasjon / samarbeid / oversikt ol. på et annet språk eller kombinert.

Legge til hvem som har referat-rollen i prosjektskissen. Hvem som har rollen følge fornavn, start med Alexander.

Møte 29.01.2021

Hvem: Veileder, Frode Haug

Hvor: Teams

Agenda: Gå gjennom rapporten for å se etter forbedringene, tips fra Frode angående rapporten.

Hovedpunkter:

Rapporten:

- Resultat mål er for teknisk, flytte punktene inn i 2.3 - Ferdig
- Fagområde - Samme volum som før, men for teknisk? - Ferdig
- Avgrensing – Inn i 2.3, skal handle om å snevre inn fagområdet til det området vi skal jobbe i, og lede videre til oppgavebeskrivelse. Kan være samme volum som fagområde
- Oppgavebeskrivelse – Mer punkter? + Legg til det fra tidligere avsnitt
- Risikoanalyse – fargelegge rutene etter risiko/sannsynlig - Ferdig
- Milepæler og Beslutningspunkter – mangler beslutningspunkt, statusrapport?

Møte 02.02.2021

Hvem: Veileder, Frode Haug

Hvor: Teams

Agenda: Gå gjennom rapporten for å se etter forbedringene, tips fra Frode angående rapporten.

Hovedpunkter:

Rapporten:

- Skrivefeil omfang.
- Fagområdet er prosjektledelse.
- Ikke noe teknisk. Fagområdet er adminsitrerering rundt prosjektet, kommunikasjon, oversikt over status.
- Fagområdet er slik vi skal jobbe. Ikke det vi skal jobbe med. Det prosjektet handler om.
- Avgrensningen er kommunikasjon med oppdragsgiver/leverandør.
- Bare hva som skal lages! Ikke noe om hvordan.
- Utviklingsmetodikk. Ikke relevant, Burde gjøre om til.
- Baktanken……
- Backlog. Viktig til sprintene.
- Planlegge og fremstille sprintene på en oversiktlig måte i rapporten.

Sprint Planning, Møte 03.02

Hvem: Gruppen

Sted: TietoEvry Brumunddal

Agenda: Planlegge neste sprint. Delegere oppgaver og komme frem til mål for neste sprintperiode.

Hovedpunkter:

Kravspesifikasjon.

- Use Case.
- Product Backlog.
- Operasjonelle krav.
- Sikkerhetskrav.
- Domenemodell.
- 15.02, prøver å få det i havn til slutten av uken, 07.02.

Spørsmål til oppdragsgiver.

- Fagområde.
- Sikkerhetskrav.
- Operasjonelle krav.

Endringer på omfang, kap.2 i prosjektskisse.

Sette opp server og koble til database.

Lage velkommenside og koble til autentifisering.

Koble autentifisering til server.

Didrik:

- 5 dager, sette opp server og database.

Timian:

- 3 dager, fikse autentifisering.

Alexander:

- 1 dag, Use Case
- 1 dag, sikkerhetskrav og operasjonelle krav.
- 1 dag, design.

Kristoffer:

- 1 dag, Lage nettside.
- 1 dag, Skrive ferdig omfang og noe kravspesifikasjon.
- 1 dag, design.

Alle:

- Mandag og tirsdag, koble sammen komponenter.

Neste sprint.

- Statusrapport. 17.02

Møte 09.02.2021

Hvem: Veileder, Frode Haug

Hvor: Zoom

Agenda: Gå gjennom rapporten for å se etter forbedringene, tips fra Frode angående rapporten. Se på ting fremover.

Hovedpunkter:

Fullstendig rapport:

- Kapittel 1 innledning (kap 2 i prosjektskisse)
- Kapittel 2 kravspec.
- Kapittel 3 design
- Kapittel 4 hvordan det er implementert, verktøy osv.

Litt om statusrapport:

- Statusrapport.
- Kontakt 5 i statusrapport. Oppdragsgiver!!

Sprint Evaluating + Sprint Planning, Møte 10.02

Hvem: Gruppen

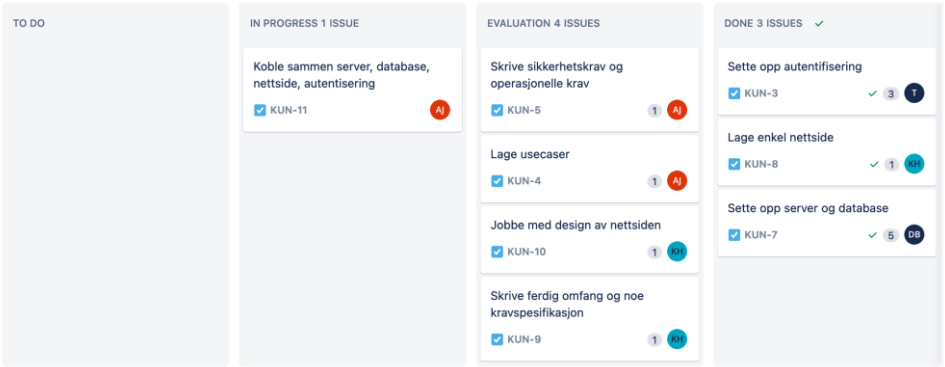
Sted: TietoEvry Brumunddal

Agenda: Se på forrige sprint og oppsummere. Planlegge neste sprint. Delegere oppgaver og komme frem til mål for neste sprintperiode.

Hovedpunkter:

Sprint Evaluation.

Back log og evaluering av forrige sprint.



- Har fått gjort det meste. Mangler noe diskusjon med oppgavegiver om kravspesifikasjon og fagområde. Har ikke koblet server og database til nettside og authentifisering enda.
- Gruppas totale evaluering: God mengde arbeid, og god utførelse. Må legge av tid til å evaluere de deler med oppgavegiver til neste sprint.

Sprint planning.

Planning poker.

- Godkjenne og gå igjennom use caser. 3 timer - 1. 5 timer - 3. Etter diskusjon 5 timer.

- Fylle opp backlog for user stories. 3 timer - 2. 5 timer - 2. Etter diskusjon 5 timer.

- Hente informasjon fra database og vise dette på nettsiden. 8 timer - 1. 13 timer - 3. Etter diskusjon 13 timer.

- Se på design og komme med utkast. 8 timer - 4. Etter diskusjon 8 timer.

- Databasemodellering. 5 timer - 3. 8 timer - 1. Etter diskusjon 8 timer.

- Navn. 5 timer - 1. 8 timer - 2. Etter diskusjon 8 timer.

Møte 10.02.2021

Hvem: Oppgavegiver

Sted: Tieto Evry Brumunddal

Agenda: Se igjennom rapport og diskutere kravspesifikasjon med oppgavegiver. Komme med statusrapport og få besvart generelle spørsmål.

Hovedpunkter:

- Begrenset tilgang ved brukerroller. Rollebasert tilgang for de forskjellige. Kan legges til under punkt 2.
- Sertifilkat på tjenesten.
- Azure authentifisering for pålogging til tjenesten.

Sikkerhetskrav.

- Https. Sertifilkat på tjenesten. Gjelder dette også for serverless?
- Rollebassert tilgang. Begrenset tilgang.
- Azure authentifisering.

- Ved utdypning.
- Administrator tildeler tilganger.
- Bruker pålogget, får data fra azure ad til egen database for å få oversikt over roller, en admin må kunne gi tilganger inne, lage noe use caser på dette. Flyten på første

gang man logger inn og hva som skjer. Hvilke rettigheter?

- Key vault i azure. Nøkler og andre passord blir lagret i key vault.

Operasjonelle krav.

- Systemet skal skaleres etter bruk. Serverless og hvorfor? Hvor mye nedetid. 99,9 oppetid i arbeidstiden. 8-16 hverdager.
- Hvordan defineres svartid. Mellom 1 og 3 sekunder. Burde klare 1 sekund? Tre sekunder er litt lenge.

MongoDB utkast.

- Holder med 1 kontakt per kunde
- Et kommentarfelt.
- Mailadresse er forhåndsutfylt
- To linker, svar eller bekreft.
- Mail sendes til kontaktperson.
- Beskrivelsesfelt hos supplier.

Sprint Evaluating + Sprint Planning, Møte 17.02

Hvem: Gruppen

Sted: TietoEvry Brumunddal

Agenda: Se på forrige sprint og oppsummere. Planlegge neste sprint. Delegere oppgaver og komme frem til mål for neste sprintperiode.

Hovedpunkter:

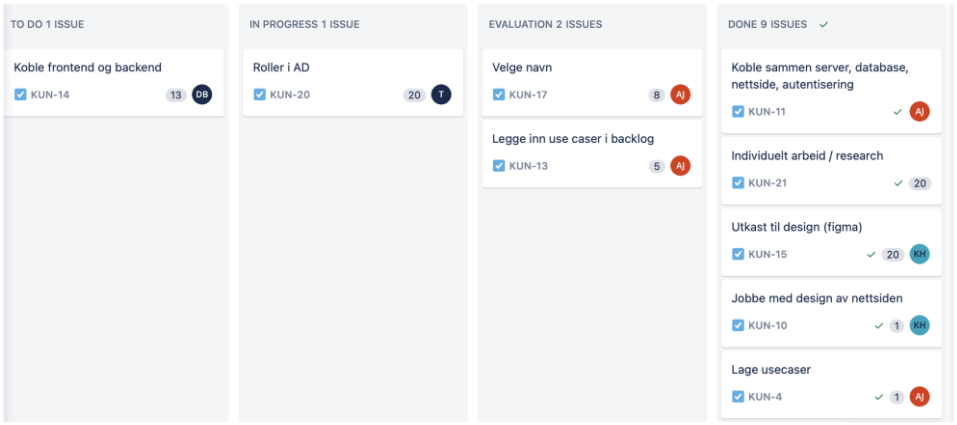
Sprint Evaluation.

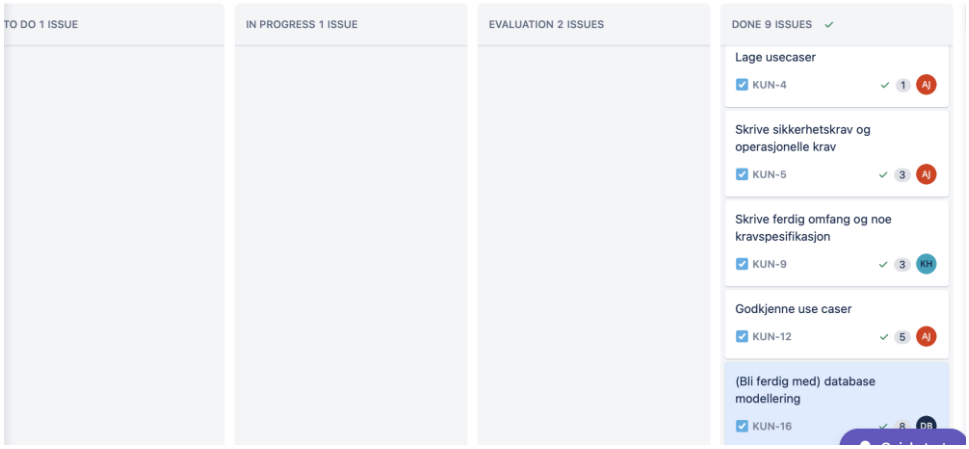
- **Sprintevaluering**

Sprint 02 har gått til utkast av design, research på hvordan koble sammen komponenter, databasemodellering, kravspesifikasjon og ferdiggjøre usecaser.

Det ble ikke fullført kobling av komponenter. Burde sette av en dag til hvor det blir utført samarbeid ved fysisk oppmøte. Uten om dette ble det meste andre uført.

Back log.





Gruppens tanker etter sprinten.

- Må legge opp til mer oppnåelige og klare sprintmål til neste sprint for å sammen jobbe mot et resultat i løpet av sprinten.
- Sette opp minst to dager i tillegg til sprintdag hvor vi jobber sammen i løpet av perioden. Mandager og fredager klokken 09.00.

Sprint Planning.

Hovedmål til neste uke. Lage en startside med mulighet til login og autentifisering med tilpasset dashboard.

- Research. Hvordan beskytte backend. 5, 8, 13, 21. Ble enige om 8.
- Sende tokens fra front til back. 3, 3, 5, 5. Ble enige om 5.
- Enkel design av side. 2, 2, 3, 3. Ble enige om 3.
- Implementasjon av siden. 8, 8, 8, 5. Ble enige om 8.
- Legge inn informasjon i DB. 1, 1, 1, 2. Ble enige om 1.
- Enkel design av dashboard. 5, 5, 8, 8. 2, 2, 3, 5. Ble enige om 3.
- Implementere websiden. 5, 12, 13, 13. 8, 8, 8, 13. Ble enige om 8.
- Hente informasjon fra databasen. 5, 5, 5, 8. Ble enige om 5.
- Autorisering i backend. 5, 5, 8, 8. Ble enige om 8.

Møte 17.02.2021

Hvem: Oppgavegiver

Sted: Tieto Evry Brumunddal

Agenda: Få oppklart ting om database, funksjonaliteter og tilganger, og andre spørsmål ved uklarheter.

Hovedpunkter:

- Admin writer og Admin reader
- Ansatt. Kan se egne kunder.

- Kunden skal ikke trenge å se kontaktperson.
- Kunden bare ha tilgang til egen informasjon.
- Bare tekstfelt for å vise til hvor informasjon viser.
- Skal applikasjonen være på norsk og engelsk? Hadde vært fint å få implementert.
- Kundeavtaler. Checkbox om det er lagt til avtaler. Kan huke av om den har avtale, med et tekstfelt/kommentar/agreement comment under til hvor det er.

Møte 23.02.2021

Hvem: Veileder, Frode Haug

Hvor: Zoom

Agenda: Høre om statusrapport er godkjent. Høre noe om forventet fremgang per dags dato ved rapport.

Hovedpunkter:

Statusrapport så bra ut. Noen spørsmål fra Frode, men helt fint.
Rapportskriving er ikke nødvendig å starte på før lynkurset i starten av mars.

Sprint Evaluating + Sprint Planning, Møte 24.02

Hvem: Gruppen

Sted: TietoEvry Brumunddal

Agenda: Se på forrige sprint og oppsummere. Planlegge neste sprint. Delegere oppgaver og komme frem til mål for neste sprintperiode.

Hovedpunkter:

Autentisering / login

Attach | Add a child issue | Link issue

Beskrivelse

Add a description...

Child issues

100% Done

~~KUN-47~~ Research: hvordan beskytte backend	8		FERDIG
~~KUN-48~~ Sende tokens fra front til back	5	T	FERDIG
~~KUN-49~~ Enkel design av side	3	KH	FERDIG
~~KUN-50~~ Implementasjon av siden	8	AJ	FERDIG
~~KUN-51~~ Legge inn informasjon i DB	1		FERDIG

Se dashboard

Attach | Add a child issue | Link issue

Beskrivelse

Add a description...

Child issues

75% Done

~~KUN-43~~ Enkel design av startside	3	AJ	FERDIG
~~KUN-44~~ Implementere websiden	8	KH	FERDIG
KUN-46 Display informasjon i dashboard	5	AJ	UNDER ARBEID
~~KUN-52~~ Autentisering i backend	8	T	FERDIG

Notater fra sprint:

Bra delegert oppgaver. Alle har jobbet hele tiden. Nesten fullstendig ferdig med målene for sprinten, mangler å display data fra databasen pga en sen bug, og å legge inn denne informasjonen i databasen.

Forbedringspotensiale:

- Komme til riktig tid og ha med mat. Mye forsinkelser, og ofte noen som må dra tidlig pga ikke har med mat.
- Bestille rom i god tid.
- Registrere timer I Clockify.

Planlegge neste sprint

- Sette opp møte med Jan Fredrik for å diskutere første innlogging og roller i AD.
- Tags, hvordan søke etter de? Tekst eller drop down. Kombinerer det?

Backlog og planning poker for neste sprint.

Nye issues:

Se kundeside (Enig om 26):

- Design av side (FIGMA). 1, 1, 3, 3. Ble enige om 1 time.
- Implementasjon av side. 1, 13, 13, 21. 8, 13, 13, 13. Ble enige om 13.
- Funksjon for å legge til kunde. 3, 5, 5, 13. 1, 3, 3, 3. Ble enige om 3.
- Funksjon for å hente kunde. 1, 1, 3, 3. Ble enig om 1.
- Begrense tilgang for kunde. 3, 3, 8, 8. 5, 8, 8, 8. Ble enige om 8.

Første pålogging (Enig om 1):

- Research og møte om roller. 1, 3, 3, 5. 1, 1, 3, 3. Ble enig om 1.

Gi rettigheter til ny innlogget:

- Design side. 1, 3, 3, 3. Ble enige om 3.
- Implementere side. 8, 13, 13, 13. Ble enige om 13.
- Funksjon for å endre rettigheter ved bruker. 1, 3, 5, 5. Ble enige om 3.
- Research og diskusjon om rettigheter. 1, 3, 5, 8. Ble enige om 3.

Legge til beskrivelse ved Use Case. 1, 1, 3, 3. Ble enige om 3.

Søke etter tags:

- GUI søkeboks. 1, 1, 3 , 5. Ble enige om 1.
- Research og diskusjon. 3, 3, 5, 5. Ble enige om 3.
- Søke i database. 1, 3, 3, 3. Ble enig om 3.
- Hente søkeresultatet fra database. Ble enig om 1.
- Display søkeresultat. 1, 3, 5, ?. Ble enig om 3.

Research på mail. 3, 5, 5, 5. Ble enig om 5.

Logodesign. 5, 5, 8, 8. Ble enig om 8.

- Velge farge. 3.

Definere kodestadard, evt. en addon. 5, 5, 8, 8. Ble enige om 5.

Testing:

- Research på hvordan. 5, 5, 8, 8. Ble enige om 8.
- Implementere tester for denne ukens sprint. 5, 5, 8, 13. Ble enige om 8.

Kommentere kode med docstrings. 1, 3, 3, 5. 3, 5, 5, 8. Ble enige om 5.

Flyttet til neste sprint: Lese igjennom en full rapport og notere. 5, 8, 8, 13. Ble enige om 13.

Møte 24.02.2021

Hvem: Jan Fredrik

Hvor: Teams

Agenda: Noen spørsmål angående roller og hvordan dette implementeres. Noe tillegg.

Hovedpunkter:

- Tennent, navn og e-post. Ikke hvilke kunder.
- Knytter kunder i applikasjonen.
- Alternativ, oppretter grupper i AD.
- Kunder logger på. Evt få gjestebruker. For å logge på. Jan Fredrik skal høre med Guro. Burde kunne løses med gjestebruker.
- Tags er cirka 10. Definert på forhånd. Legge til en tag, en tekst. Skal være mulig å legge til nye tags etterhvert.
- Navn. Synes Tekra hadde god schwong.

Sprint Evaluating + Sprint Planning, Møte 03.03

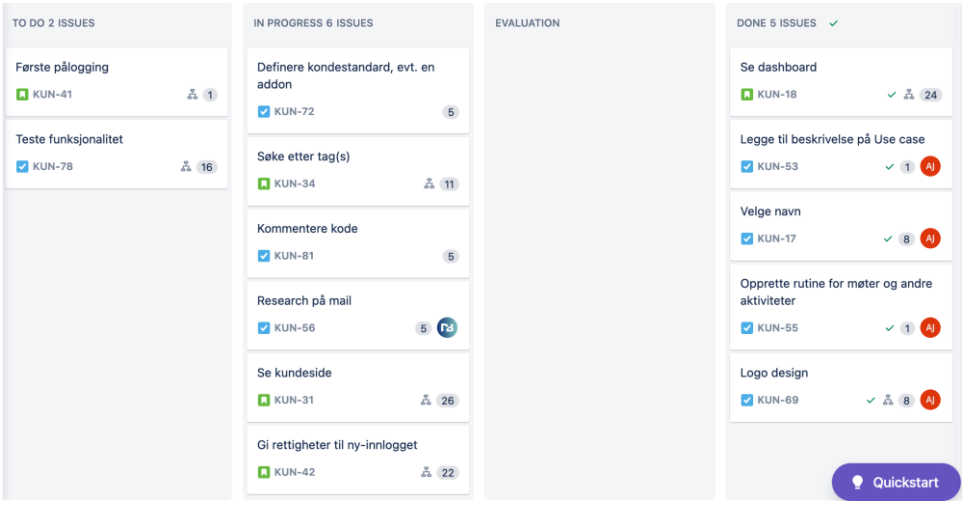
Hvem: Gruppen

Sted: TietoEvry Brumunddal

Agenda: Se på forrige sprint og oppsummere. Planlegge neste sprint. Delegere oppgaver og komme frem til mål for neste sprintperiode.

Hovedpunkter:

Evaluering.



Denne uken ble det mye arbeid, og noen utilsiktede hendelser ved planleggingen. Seminar hele mandag og innlevering av oblig i TØL faget. Dette resulterte i en mindre effektiv sprint. Allikevel er det meste av funksjonalitet gjort. Det er mindre ting som gjenstår ved de forskjellige sprintene.

Målet for sprinten er ikke godkjent og vi feilestimerte denne sprinten, men allikevel er det utført god mengde arbeid.

Sprint Planning 03.03 - 10.03.

Oppdatert fra forrige uke.

Definere kodestandard, evt en addon. 1, 1, 1, 1. Ble enige om 1.

Søke etter tags. Samme.

Kommentere kode. 3, 3, 3, 3. Ble enige om 3.

Research på mail. Samme.

Se kundeside. Samme.

- Lagt til ny childissue Hent Alt Om Kunde. 1, 3, 3, 3. Ble enige om 3.
- Lagt til ny childissue. Research på url. 3, 3, 3, 5. Ble enige om 3.

Gi rettigheter til ny-innlogget. Samme. 1, 3, 3, 5. Ble enige om 3.

Første pålogging. Ble enige om 3.

Lese hel rapport. Samme.

Teste funksjonalitet. Samme.

Sende ny mail. Neste sprint.

- Designe side. 1, 1, 1, 3. Ble enige om 1.
- Implementere side. 8, 13, 13, 13. Ble enige om 13.
- Funksjon for å sende mail. 8, 8, 8, 8. Ble enige om 8.

Sende mail til flere. Neste sprint.

Autorisering. Rolle.

Sprint Evaluating + Sprint Planning, Møte 10.03

Hvem: Gruppen

Sted: TietoEvry Brumunddal

Agenda: Se på forrige sprint og oppsummere. Planlegge neste sprint. Delegere oppgaver og komme frem til mål for neste sprintperiode.

Hovedpunkter:

Evaluering.

I følge gantt-diagrammet skulle vi ha ferdigstilt en prototype i løpet av dagen, men er ikke ferdigstilt enda. Skal jobbe videre med dette inn I neste sprint.

Evalueringen falt på at det ble litt mye å gjøre, og samtidig ble det vanskelig å gjennomføre det som ble planlagt tidligere i Gantt-diagrammet.

Sprint Planning.

Design av ny kundeside/redigere. 1, 1, 1, 3. Ble enige om 1.

Implementere design for kundeside. 3, 3, 5, 8. Ble enige om 3.

Implementere funksjonalitet for ny kunde. 5, 5, 8, 8. Ble enige om 8.

Design av side for å legge nye leverandører. 1, 1, 1, 1. Ble enige om 1.

Implementere design av leverandørside. 3, 3, 3, 5. Ble enige om 3.

Implementere funksjonalitet for leverandør. 5, 5, 5, 8. Ble enige om 5

Motta mail. 8, 8, 8, 13. Ble enige om 8.

Rediriect etter login. 5, 5, 5, 5. Ble enige om 5.

Møte 16.03.2021

Hvem: Veileder, Frode Haug

Hvor: Zoom

Agenda: Spørsmål rundt introduksjon til rapporten, og litt om nåværende status ved prosjektet.

Hovedpunkter:

Så bra ut, må endre noe på:

- Avgrensning
- Formål
- Målgruppe

Planlegger et utkast av rapport til etter påske.

Sprint Evaluating + Sprint Planning, Møte 17.03

Hvem: Gruppen

Sted: TietoEvry Brumunddal.

Agenda: Se på forrige sprint og oppsummere. Planlegge neste sprint. Delegere oppgaver og komme frem til mål for neste sprintperiode.

Hovedpunkter:

Evaluering.

Litt middels sprint. Ikke gjort like mye som ble planlagt, men tid tok litt ekstra tid.

Lage API - dokumentasjon. Enige om 5.

Slette leverandør. Enige om 3.

Slette kunde. Enige om 3.

Slette ansatte. Enige om 3.

Møte 24.03.2021

Hvem: Oppdragsgiver, Jan Fredrik Gundersen

Hvor: Teams

Agenda: Status og spørsmål rundt uklarheter.

Hovedpunkter:

God tilbakemelding. Virker å være på samme bølgelengde. Noen tips underveis.

- letsEncrypt. Må ha —>CertBot. Får generert gratis sertifikat.
- Certify SSL manager. Certifyweb.

Sprint Evaluating + Sprint Planning, Møte 24.03

Hvem: Gruppen

Sted: Teams.

Agenda: Se på forrige sprint og oppsummere. Planlegge neste sprint. Delegere oppgaver og komme frem til mål for neste sprintperiode.

Hovedpunkter:

Evaluering.

Fikk gjennomført mye av de største oppgavene. Noen av oppgavene trenger finpuss for å ferdiggjøres. Har planlagt neste sprint til etter påske.

Sprint planning.

Satt opp rapportskriving som fokus gjennom påsken og la vekt på å gjennomføre resterende oppgaver fra tidligere sprinter. Legger til rette for å ikke jobbe gjennom hele påskeferien.

Sprint Evaluating + Sprint Planning, Møte 07.04

Hvem: Gruppen

Sted: Teams.

Agenda: Se på forrige sprint og oppsummere. Planlegge neste sprint. Delegere oppgaver og komme frem til mål for neste sprintperiode.

Hovedpunkter:

Evaluering:

Sprinten har denne perioden vart over påsken hvor all har hatt tid for ferie. Vi har fokusert på at alle skriver på rapporten over påskeferien, og det har blitt utfylt. Utenom dette har sprinten blitt behandlet likt som tidligere sprinter uten mye ekstra arbeidsoppgaver.

Arbeidsoppgavene ble i hovedsak utført, og det har vært en vellykket sprint.

Sprint planning:

Kunder får egen profil når de logger på for første gang.

- Estimering. 5, 5, 8, 8. Ble enige om 5.

Legge til testbruker. (Kunde)

- Estimering. 3, 3, 3, 3. Ble enige om 3.

Legge til søkefunksjon i rettighetside.

- Estimering. 1, 1, 1, 1. Ble enige om 1.

Feilmelding i frontend når ikke tilgang på kunde/leverandør.

- Estimering 1, 3, 3, 5. En gang til. 3, 3, 3, 3. Enig om 3.

Feilmelding når applikasjonen ikke kan hente data.

- Estimering 1, 1, 1, 1, Ble enige om 1.

Endre språk.

Oversette side

- Estimering. 3, 5, 5, 5. 3, 3, 3, 3. Enige om 3.

Endre språk og finne bibliotek.

- Estimering. 5, 5, 5, 5. Enige om 5.

SSL sertifikat.

- Estimering. 3, 3, 3, 5. Enige om 3.

Sprintmål. Ferdigstille funksjonalitet, pynte litt på design, og skrive en del på rapport og levere til Frode for evaluering innen torsdag 08.04.

Veiledningsmøte med Frode, 09.04

Hvem: Frode

Agenda: Gjennomgå utkast til rapporten og få tilbakemelding på forbedringer.

Hovedpunkter:

Bruke malen fra Word

Akronymer —> Legge i vedlegg A.

Legge inn målene etter målgruppe. Kan også henvise.

Introduksjon —> En skrives kursivt.

Brukergrensesnitt —> Skrive kapittelet tidligere. Kan skrives før Systemdesign.

Deployment norsk —> Installasjon.

Kap 2.

- Lavnivå / Høynivå. Hva er hva?
- Flyt.cloud —> Enten introdusere tidligere, eller kutte helt ut i UML-diagrammet.
- Visuelt skal hele figuren være rett under teksten.

Kap 3.

- Kan skrive en liten del om scrum, og kan være å si noe, men er ikke nødvendig.
- Utvidede nøkkelord om hver sprint. Kulepunkter 2, 3 eller 4.

Kap 6.

- Bruke skjermbilder.
- Pattern: Mønster / mal / prototype.

Kilde fotnote, legge til sist besøkt.

Referanseliste, kilder som blir brukt flere ganger.

Testingkapittel. Mange lager dette syltynt. Viktig å utdype og forklare godt!!

- Få testet ved noen brukergrupper. Stort pluss.

Sprint Evaluating + Sprint Planning, Møte 14.04

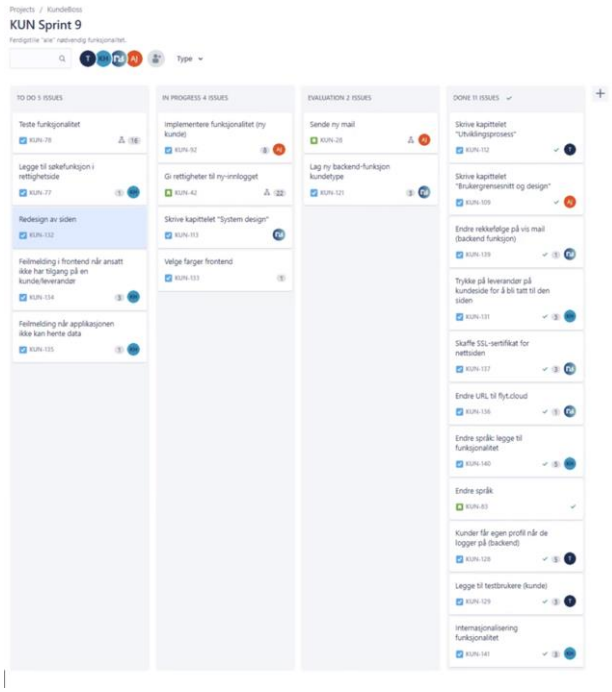
Hvem: Gruppen

Sted: Teams.

Agenda: Se på forrige sprint og oppsummere. Planlegge neste sprint. Delegere oppgaver og komme frem til mål for neste sprintperiode.

Hovedpunkter:

Sprint evaluering.



- God innsats. Har vært noe sykdom som har gjort det vanskeligere å møtes alle sammen. Alt i alt, en godkjent sprint.

Sprint planning.

Oppdatere database

- Research. Enige om 5.
- Implementasjon backend. Enige om 5.
- Design I firma. Enige om 1.
- Implementere i frontend. Enige om 5.
- Skrive Deployments. Enige om 12.

Skrive Implementasjon.

- Backend. Enige om 8.
- Frontend. Enige om 8.

Sprint Evaluating + Sprint Planning, Møte 21.04

Hvem: Gruppen

Sted: Teams.

Agenda: Se på forrige sprint og oppsummere. Planlegge neste sprint. Delegere oppgaver og komme frem til mål for neste sprintperiode.

Hovedpunkter:

Sprint evaluation.

- Fikk ikke gjennomført slik vi ønsket, men hadde også større deloppgaver som omhandlet å skrive ferdig kapittel til rapporten. Fikk gjennomført en del, men ikke ferdigstilt alle oppgavene.

Sprint Planning.

- Neste sprint er planlagt å være en skrivesprint. Til mandag skal vi levere utkast til rapport til Frode. Går igjennom alt og skriver implementasjon og deployment.

Sprint Goals.

- Skrive ferdig utkast til rapport.
- Starte redesign av siden.
- Fullføre 'implementasjon'.

Sprint Evaluating + Sprint Planning, Møte 28.04

Hvem: Gruppen

Sted: Teams.

Agenda: Se på forrige sprint og oppsummere. Planlegge neste sprint. Delegere oppgaver og komme frem til mål for neste sprintperiode.

Hovedpunkter:

Sprint evaluering.

- Brukt mye tid på rapportskriving de siste dagene, pga innlevering av utkast til Frode. Gruppen jobbet godt med oppgavene som ble satt opp, men noen av disse, som kapitlene og revideringen av design og responsivitet tok mye lengre tid enn antatt. Mye jobb, men gruppen evaluerer sprinten til å være godt gjennomført.

Sprint planning.

- Neste sprint går på å gjennomføre revideringen av design og repsonsivitet slik som forrige gang. Gjøre endringer ved rapporten etter tilbakemelding fra Frode. Har lagt til diverse oppgaver fra backlog.

Nye oppgaver:

- Ferdigstille logo - Alle enige om 1.
- Fikse på ting Frode nevner ved rapport - Enige om 13.
- Bruker blir sendt til riktig side ved innlogging - Enige om 8.
- Brukertest - Enige om 5.
- Unittesting frontend - Enige om 13.
- Unittesting backend - Enige om 21.

Tema for sprint: Fortsette forrige sprint. Design, skriving og testing.

Sprintmål: Bli ferdig med design og testing. Skrive ferdig første 6 kapitler.

Veiledningsmøte med Frode, 29.04

Hvem: Frode

Agenda: Gjennomgå utkast til rapporten og få tilbakemelding på forbedringer. Andre utkast

Hovedpunkter:

Kap 1. Formål

Nytt formål - Legge til hvorfor.

“Formålet med prosjektet er å utvikle en forbedring av nåværende praksis omkring oppdragsgivers system rundt kundebehandling, organisering, og oversiktlighet for å effektivisere deres arbeidsprosess”

Kap 2. Lavnivå vs Høynivå use case.

Stor bokstav ved begge ord ved Scrum?

Kap3. Hva menes med space mellom rammene så de ikke fyller siden? Er det space inne i boksen, eller mer space mellom rammene? Fylle ut resten av den tomme siden med space i mellom.

Fylle siden slik at ikke halve siden er tom. Boksene, prøve fylle sidene med enten mindre tekst eller mer tekst. Skal se bra ut på i dokumentet.

Hadde det vært bedre å skrive kortere og heller vedlegge sprintinfo?

Kapittel 6. Hva skal være med hvis ikke utgivelse skal være med? Kan ha CI/CD i implementasjon for å vise hvordan det er gjort og henvise til dette i kapittel 6.

Kapittel 2 sikkerhetskrav - Definere hvordan dette løses senere i rapporten!!

Få klarert kilder.

- Dersom større kilder vi har brukt mye, kildeliste. Hvis forklaringer ved enkelt begreper eller engangs kilde, fotnote.

Frode mener vi ligger ann til å skrive en bra rapport!

Veiledningsmøte med Frode, 04.05

Hvem: Frode

Hvor: Zoom

Agenda: Svare på spørsmål angående kapittel 5 og 6.

Hovedpunkter:

Teknisk design:

- Skal være videre fra kravspesifikasjon og før detaljert implementering med henvisning til kode.
- Ikke for detaljert.

Implementasjon:

- Implementasjon skal ikke inneholde alt.
- Eksempler på kode, moderat vanskelighetsgrad.
- Referere til kode ved .zip-fil.

Sende mail med kap.5 til Frode for evaluering.

Sprint Evaluating + Sprint Planning, Møte 05.05

Hvem: Gruppen

Sted: Teams.

Agenda: Se på forrige sprint og oppsummere. Planlegge neste sprint. Delegere oppgaver og komme frem til mål for neste sprintperiode.

Hovedpunkter:

Evaluering:

Fikk gjennomført de største aspektene ved sprinten. Brukte mer tid på rapport enn først antatt, og hadde to evalueringsmøter med veileder. Fikk gjennomført det viktigste, men har lagt til mye ved sprinten. Underestimert rapportskriving, og design av siden. Alt i alt, godt arbeid og mye ble gjort.

Sprint planning:

Sprintmål: Rette på feil og småjusteringer etter brukertest. Skrive ferdig til og med kapittel 7 i rapport. Siste design av siden.

Møte 07.05.2021

Hvem: Oppdragsgiver, Guro Storlien Evensen og Jan Fredrik Gundersen

Hvor: Brumunddal

Agenda: Nåværende status og presentasjon av systemet.

Hovedpunkter:

Presenterte tilnærmet ferdigstilt system, med noen mindre mangler ved design, for oppdragsgiver for å få tilbakemelding dersom noe skulle endres, fjernes eller legges til i systemet for å tilfredsstille deres krav. Fikk tilbakemelding om at alt fungerte slik de hadde sett for seg, og vi har utført de kravene de stilte ved kravspesifikasjon.

Fikk en tilbakemelding på varsler dersom en kunde har ny mail ved seg. Dette er ikke tidligere blitt definert, og vil derfor nedprioriteres som tilleggsfunksjonalitet ved systemet, men gruppen vil se om det er muligheter for å implementere dette om det er tid mot slutten.

Fikk noen tips til utarbeiding av rapport. Blant annet, henvise til gode kilder i kildeliste, og vise til gode bildeeksempler med forklaring av hvordan brukergrensesnitt fungerer.

Ble invitert til workshop for demonstrasjon av hvordan systemet er implementert til videre utvikling av systemet.

Sprint Evaluation, Møte 12.05

Hvem: Gruppen

Sted: Teams.

Agenda: Se på forrige sprint og oppsummere.

Hovedpunkter:

Sprint evaluering.

- Fått gjennomført oppsatte tasks. Har en uke på finpuss, og vil skrive ferdig rapporten i løpet av de neste dagene. Alt i alt, en godt gjennomført sprint og alle arbeidet godt gjennom uken.

Veiledningsmøte med Frode, 12.05

Hvem: Frode

Hvor: Zoom

Agenda: Vurdering av kapittel 5 og 6, og videre utarbeiding av diskusjon, konklusjon og sammendrag.

Hovedpunkter:

Synes vi har gjort det greit til nå, fortsette i samme kjøret.

Bindestrek ved “API-et".

Ta heller med for mye enn for lite.

Diskusjon - Hva kunne vært gjort annerledes? Grafisk teknisk. Diskutere hva vi har gjort, og hva vi kunne gjort annerledes med arbeidsmetodikken. 3-8 vanlig 3-5 sider.

Konklusjon - Hva vi sitter igjen med, hvordan har det gått, nådde mål ……. Cirka en halv side.

Vedlegg I

Timelogg

Navn	11.01.2021	12.01.2021	13.01.2021	14.01.2021	15.01.2021	16.01.2021	17.01.2021	Total
Alexander		04:00:00	03:00:00	06:30:00	04:00:00			17:30:00
Didrik		04:00:00	05:00:00	05:00:00				14:00:00
Kristoffer		04:00:00	02:00:00	05:00:00				11:30:00
Timian		05:30:00	02:30:00	03:00:00	01:00:00	03:30:00		15:00:00
Total		17:30:00	12:30:00	19:30:00	05:00:00	03:30:00		58:00:00

Tabell I.1: Timelogg uke 2

Navn	18.01.2021	19.01.2021	20.01.2021	21.01.2021	22.01.2021	23.01.2021	24.01.2021	Total
Alexander	03:30:00	01:00:00	06:30:00		01:30:00	04:00:00		16:30:00
Didrik	05:30:00	06:00:00	06:30:00			04:00:00		22:00:00
Kristoffer	01:30:00	05:45:00	07:00:00			01:00:00		15:45:00
Timian	03:00:00	04:00:00	07:00:00		07:00:00	01:00:00		22:00:00
Total	13:30:00	16:45:00	27:00:00		08:30:00	10:00:00		76:15:00

Tabell I.2: Timelogg uke 3

Navn	25.01.2021	26.01.2021	27.01.2021	28.01.2021	29.01.2021	30.01.2021	31.01.2021	Total
Alexander	05:00:00	05:00:00	08:30:00	04:30:00	02:00:00		06:00:00	31:00:00
Didrik	05:00:00	05:00:00	08:30:00	07:00:00	02:00:00	01:00:00		28:30:00
Kristoffer	06:00:00	04:30:00	07:15:00	02:00:00			05:00:00	24:45:00
Timian	05:00:00	06:00:00	08:30:00	04:00:00			03:00:00	26:30:00
Total	21:00:00	20:30:00	32:45:00	17:30:00	04:00:00	01:00:00	14:00:00	110:15:00

Tabell I.3: Timelogg uke 4

Navn	01.02.2021	02.02.2021	03.02.2021	04.02.2021	05.02.2021	06.02.2021	07.02.2021	Total
Alexander	01:00:00	09:00:00	09:30:00	05:00:00	06:00:00			30:30:00
Didrik		00:45:00	10:00:00	05:30:00	04:00:00	04:00:00	03:00:00	27:15:00
Kristoffer		04:00:00	08:30:00	05:00:00	05:00:00			22:30:00
Timian	01:30:00	01:45:00	08:30:00	10:30:00		04:00:00	08:15:00	34:30:00
Total	02:30:00	15:30:00	36:30:00	26:00:00	15:00:00	08:00:00	11:07:52	114:45:00

Tabell I.4: Timelogg uke 5

Navn	08.02.2021	09.02.2021	10.02.2021	11.02.2021	12.02.2021	13.02.2021	14.02.2021	Total
Alexander	06:00:00		08:00:00	03:00:00	05:00:00			22:00:00
Didrik	06:00:00		07:15:00	07:00:00		06:00:00		26:15:00
Kristoffer	06:00:00	05:00:00	07:00:00		05:00:00			23:00:00
Timian	02:00:00	08:30:00		04:00:00				14:30:00
Total	20:00:00	05:00:00	30:45:00	10:00:00	14:00:00	06:00:00		85:45:00

Tabell I.5: Timelogg uke 6

Navn	15.02.2021	16.02.2021	17.02.2021	18.012.2021	19.02.2021	20.02.2021	21.02.2021	Total
Alexander	03:30:00	04:00:00		12:30:00	04:00:00	03:00:00		27:00:00
Didrik		03:30:00	07:15:00	04:00:00	05:00:00	02:00:00	04:00:00	25:45:00
Kristoffer			08:30:00	07:00:00	05:00:00			20:30:00
Timian	04:00:00		08:00:00		07:00:00			19:00:00
Total	07:30:00	07:30:00	23:45:00	23:30:00	21:00:00	05:00:00	04:00:00	92:15:00

Tabell I.6: Timelogg uke 7

Navn	22.02.2021	23.02.2021	24.02.2021	25.012.2021	26.02.2021	27.02.2021	28.02.2021	Total
Alexander	03:00:00	05:00:00	08:00:00	03:00:00	08:30:00		02:00:00	29:30:00
Didrik	06:00:00	08:00:00	08:00:00		08:00:00		03:00:00	33:00:00
Kristoffer	04:00:00	05:00:00	08:00:00		08:00:00			25:00:00
Timian	05:15:00	02:30:00	09:00:00		07:15:00			24:00:00
Total	18:15:00	20:30:00	33:00:00	03:00:00	31:45:00		05:00:00	111:30:00

Tabell I.7: Timelogg uke 8

Navn	01.03.2021	02.03.2021	03.03.2021	04.03.2021	05.03.2021	06.03.2021	07.03.2021	Total
Alexander	05:00:00	07:00:00	06:00:00	02:00:00	05:00:00			25:00:00
Didrik			11:30:00		04:00:00			15:30:00
Kristoffer	06:00:00		06:00:00		06:00:00			18:00:00
Timian	08:15:00		05:00:00		06:00:00		04:00:00	23:15:00
Total	19:15:00	07:00:00	28:30:00	02:00:00	21:00:00		04:00:00	81:45:00

Tabell I.8: Timelogg uke 9

Navn	08.03.2021	09.03.2021	10.03.2021	11.03.2021	12.03.2021	13.03.2021	14.03.2021	Total
Alexander	05:00:00	05:00:00	09:00:00		06:00:00			25:00:00
Didrik	06:00:00	07:30:00	08:00:00		01:45:00	03:00:00		26:15:00
Kristoffer	06:00:00	06:00:00	08:00:00		03:00:00			23:00:00
Timian	06:15:00	06:30:00	08:00:00	01:00:00	03:30:00			25:30:00
Total	23:15:00	25:00:00	33:00:00	01:00:00	14:15:00	03:00:00		99:45:00

Tabell I.9: Timelogg uke 10

Navn	15.03.2021	16.03.2021	17.03.2021	18.03.2021	19.03.2021	20.03.2021	21.03.2021	Total
Alexander	05:30:00		08:00:00	04:00:00	03:00:00		03:30:00	24:00:00
Didrik	03:00:00	07:00:00	07:15:00		02:00:00			19:15:00
Kristoffer	07:00:00	02:00:00	08:00:00	04:00:00	08:00:00			29:00:00
Timian	03:15:00	01:45:00	07:15:00		02:00:00	03:30:00	03:00:00	20:45:00
Total	18:45:00	10:45:00	30:30:00	08:00:00	15:00:00	03:30:00	06:30:00	93:00:00

Tabell I.10: Timelogg uke 11

Navn	22.03.2021	23.03.2021	24.03.2021	25.03.2021	26.03.2021	27.03.2021	28.03.2021	Total
Alexander	09:00:00	01:30:00	08:30:00					19:00:00
Didrik	05:45:00		06:30:00		07:15:00			19:30:00
Kristoffer	06:00:00	02:00:00	06:00:00	04:00:00				18:00:00
Timian	05:00:00		10:00:00		05:00:00			20:00:00
Total	25:45:00	03:30:00	31:00:00	04:00:00	12:15:00			76:30:00

Tabell I.11: Timelogg uke 12

Navn	29.03.2021	30.03.2021	31.03.2021	01.04.2021	02.04.2021	03.04.2021	04.04.2021	Total
Alexander		04:00:00	02:00:00		04:00:00			10:00:00
Didrik	04:00:00	05:00:00	05:00:00					14:00:00
Kristoffer	04:00:00	04:00:00	04:00:00					12:00:00
Timian	03:00:00	01:00:00	01:00:00		02:00:00			07:00:00
Total	11:00:00	14:00:00	12:00:00		06:00:00			43:00:00

Tabell I.12: Timelogg uke 13

1

Navn	05.04.2021	06.04.2021	07.04.2021	08.04.2021	09.04.2021	10.04.2021	11.04.2021	Total
Alexander	02:00:00	03:00:00	09:00:00	04:00:00				18:00:00
Didrik			07:00:00	06:00:00	07:00:00		02:00:00	22:00:00
Kristoffer		05:00:00	06:00:00	06:00:00	03:00:00			20:00:00
Timian		06:00:00	08:30:00		05:00:00		01:30:00	21:00:00
Total	02:00:00	14:00:00	30:30:00	16:00:00	15:00:00		03:30:00	81:00:00

Tabell I.13: Timelogg uke 14

Navn	12.04.2021	13.04.2021	14.04.2021	15.04.2021	16.04.2021	17.04.2021	18.04.2021	Total
Alexander	05:00:00	05:00:00	04:00:00		03:00:00		03:00:00	20:00:00
Didrik		06:00:00	06:45:00	06:30:00	04:00:00			23:15:00
Kristoffer	03:00:00	07:00:00	06:00:00	04:00:00	03:30:00			23:30:00
Timian	02:00:00	04:30:00	10:30:00		03:00:00			20:00:00
Total	10:00:00	22:30:00	27:15:00	10:30:00	13:30:00		03:00:00	86:45:00

Tabell I.14: Timelogg uke 15

Navn	19.04.2021	20.04.2021	21.04.2021	22.04.2021	23.04.2021	24.04.2021	25.04.2021	Total
Alexander	07:00:00	04:30:00	09:00:00	05:00:00				25:30:00
Didrik	06:45:00	03:30:00	06:45:00	05:45:00	03:00:00			25:45:00
Kristoffer	07:00:00	04:00:00	07:00:00	05:00:00				23:00:00
Timian	07:15:00	04:00:00	07:00:00		03:00:00			21:15:00
Total	28:00:00	16:00:00	29:45:00	15:45:00	06:00:00			95:30:00

Tabell I.15: Timelogg uke 16

Navn	26.04.2021	27.04.2021	28.04.2021	29.04.2021	30.04.2021	01.05.2021	02.05.2021	Total
Alexander	06:00:00	04:00:00	09:00:00		04:00:00		02:00:00	25:00:00
Didrik	04:00:00	04:00:00	05:45:00		05:45:00			19:30:00
Kristoffer	06:00:00	07:00:00	05:45:00	06:30:00	07:00:00		06:00:00	38:15:00
Timian	06:00:00	06:00:00	07:00:00		05:00:00			24:00:00
Total	22:00:00	21:00:00	27:30:00	06:30:00	21:45:00		08:00:00	106:45:00

Tabell I.16: Timelogg uke 17

Navn	03.05.2021	04.05.2021	05.05.2021	06.05.2021	07.05.2021	08.05.2021	09.05.2021	Total
Alexander	07:00:00	08:00:00	06:00:00	09:30:00	06:00:00		03:00:00	39:30:00
Didrik	07:15:00	06:00:00	06:00:00		06:30:00			25:45:00
Kristoffer	07:00:00	06:00:00	06:00:00	05:00:00	08:00:00	03:00:00	04:00:00	39:00:00
Timian	09:00:00	04:15:00	08:15:00		06:00:00		04:30:00	32:00:00
Total	30:15:00	24:15:00	26:15:00	14:30:00	26:30:00	03:00:00	11:30:00	136:15:00

Tabell I.17: Timelogg uke 18

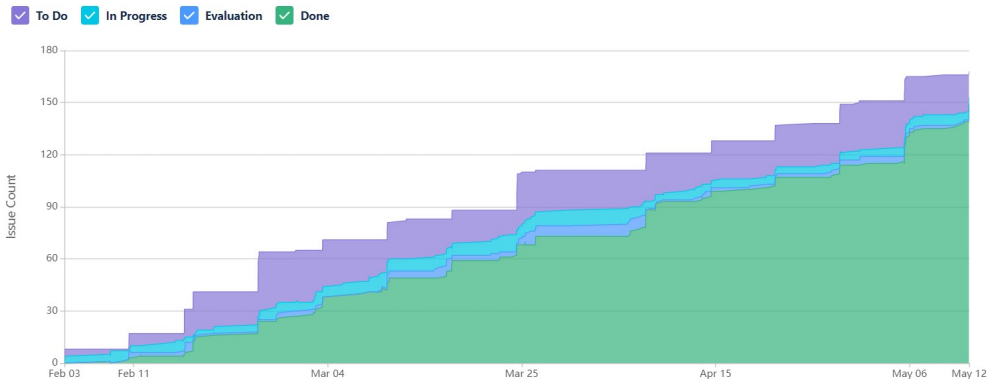
Navn	10.05.2021	11.05.2021	12.05.2021	13.05.2021	14.05.2021	15.05.2021	16.05.2021	Total
Alexander	08:30:00	08:00:00	09:30:00	10:00:00	08:30:00	02:00:00	10:30:00	57:00:00
Didrik	06:00:00	05:00:00	08:00:00	06:00:00	02:00:00	08:00:00	10:30:00	42:45:00
Kristoffer	05:00:00	10:00:00	08:00:00	10:00:00	11:00:00	11:00:00	10:00:00	65:00:00
Timian	08:00:00	04:15:00	10:30:00	08:15:00	10:00:00	09:00:00	12:00:00	62:00:00
Total	27:30:00	27:15:00	36:00:00	34:15:00	31:30:00	30:00:00	39:15:00	226:45:00

Tabell I.18: Timelogg uke 19

Vedlegg J

Jira

J.1 Graf for prosjektet



Figur J.1: Progresjon av oppgaver i Jira

J.2 Logg

Jira

Prosjekt: Flyt
Sortert etter: Opprettet synkende
1–168 av **168** som ved: **20/mai/21 2:28 AM**

T	ID	Sammendrag	Ansvarlig	Innmelder	Pri	Status	Løsning	Laget	Oppdatert	Frist
	KUN-189	Legge til mail-knapp på dashbord + 'huke av' på alle kunder	*Ikke tildelt*	Alexander A Jørgensen	↑	GJØREMÅL	*Uløst*	12/mai/21	12/mai/21	
	KUN-188	Vise om det er noe nytt i mail fra kunde	*Ikke tildelt*	Alexander A Jørgensen	↑	FERDIG	Utført	12/mai/21	20/mai/21	
	KUN-187	Notifikasjon når ny mail eller nytt svar er registrert, Backend	Didrik Bjerk	Didrik Bjerk	↑	FERDIG	Utført	09/mai/21	11/mai/21	
	KUN-186	Legge til kunde/leverandør navn på vis kunde/leverandør	Alexander A Jørgensen	Alexander A Jørgensen	↑	FERDIG	Utført	05/mai/21	11/mai/21	
	KUN-185	Trykke på logo tar deg til riktig side	*Ikke tildelt*	Alexander A Jørgensen	↑	FERDIG	Utført	05/mai/21	05/mai/21	
	KUN-184	Oversette applikasjonen	Kristoffer Haugen	Alexander A Jørgensen	↑	FERDIG	Utført	05/mai/21	12/mai/21	
	KUN-183	Fjerne rediger knapp for kunde når man kun har leserettigheter	Kristoffer Haugen	Alexander A Jørgensen	↑	FERDIG	Utført	05/mai/21	05/mai/21	
	KUN-182	Endre så admin har tilgang på alle kunder (backend)	Didrik Bjerk	Alexander A Jørgensen	↑	FERDIG	Utført	05/mai/21	05/mai/21	
	KUN-181	Endre navbar	Kristoffer Haugen	Alexander A Jørgensen	↑	FERDIG	Utført	05/mai/21	12/mai/21	
	KUN-180	Vise feilmelding når man ikke har tilgang til kunde	Kristoffer Haugen	Alexander A Jørgensen	↑	GJØREMÅL	*Uløst*	05/mai/21	12/mai/21	
	KUN-179	Fjerne console.log fra hele applikasjonen	*Ikke tildelt*	Alexander A Jørgensen	↑	UNDER ARBEID	*Uløst*	05/mai/21	12/mai/21	
	KUN-178	Legge til referanser til andre steder i rediger og info om kunde	Alexander A Jørgensen	Alexander A Jørgensen	↑	FERDIG	Utført	05/mai/21	05/mai/21	
	KUN-177	Redesign admin-side	Alexander A Jørgensen	Alexander A Jørgensen	↑	FERDIG	Utført	05/mai/21	11/mai/21	
	KUN-176	Utføre brukertester	*Ikke tildelt*	Alexander A Jørgensen	↑	FERDIG	Utført	05/mai/21	12/mai/21	
	KUN-175	Legge til kundekategorier i frontend	*Ikke tildelt*	Alexander A Jørgensen	↑	GJØREMÅL	*Uløst*	05/mai/21	12/mai/21	
	KUN-174	Sende med kontaktperson for leverandører i rediger kunde	Alexander A Jørgensen	Alexander A Jørgensen	↑	FERDIG	Utført	05/mai/21	05/mai/21	
	KUN-173	Legge inn slette-funksjon i rediger kunde/leverandør	Alexander A Jørgensen	Alexander A Jørgensen	↑	FERDIG	Utført	05/mai/21	05/mai/21	

T	ID	Sammendrag	Ansvarlig	Innmelder	Pri	Status	Løsning	Laget	Oppdatert	Frist
	KUN-172	Gjøre så man kan søke på flere ting samtidig	Kristoffer Haugen	Alexander A Jørgensen	↑	FERDIG	Utført	30/apr/21	05/mai/21	
	KUN-171	Gjøre så søk i dashbord også søker på navn	Kristoffer Haugen	Alexander A Jørgensen	↑	FERDIG	Utført	30/apr/21	05/mai/21	
	KUN-170	Legge til kontaktperson til supplier i ny kunde-funksjon	Timian	Alexander A Jørgensen	↑	FERDIG	Utført	28/apr/21	05/mai/21	
	KUN-169	Legge inn supplier.contact.name/mail/tlf	Didrik Bjerk	Alexander A Jørgensen	↑	FERDIG	Utført	28/apr/21	28/apr/21	
	KUN-168	Endre fra mail til ID i api	Timian	Alexander A Jørgensen	↑	FERDIG	Utført	28/apr/21	05/mai/21	
	KUN-167	Implementer at admin kan gi brukere forskjellige roller?	Alexander A Jørgensen	Alexander A Jørgensen	↑	FERDIG	Utført	28/apr/21	07/mai/21	
	KUN-166	Legge til types på returverdier i api	Alexander A Jørgensen	Alexander A Jørgensen	↑	UNDER ARBEID	*Uløst*	28/apr/21	20/mai/21	
	KUN-165	lage unittester backend	Didrik Bjerk	Alexander A Jørgensen	↑	FERDIG	Utført	28/apr/21	10/mai/21	
	KUN-164	Lage unittester frontend	*Ikke tildelt*	Alexander A Jørgensen	↑	EVALUATION	*Uløst*	28/apr/21	20/mai/21	
	KUN-163	Lage brukertester	Kristoffer Haugen	Alexander A Jørgensen	↑	FERDIG	Utført	28/apr/21	05/mai/21	
	KUN-162	Fikse på ting frode sier	*Ikke tildelt*	Alexander A Jørgensen	↑	FERDIG	Utført	28/apr/21	05/mai/21	
	KUN-161	Logo 100% ferdig	Alexander A Jørgensen	Alexander A Jørgensen	↑	FERDIG	Utført	28/apr/21	05/mai/21	
	KUN-160	Bruker blir sendt til riktig side når du logger på	*Ikke tildelt*	Alexander A Jørgensen	↑	FERDIG	Utført	28/apr/21	05/mai/21	
	KUN-159	Legge til error på alle forms	Alexander A Jørgensen	Alexander A Jørgensen	↑	GJØREMÅL	*Uløst*	25/apr/21	12/mai/21	
	KUN-158	Legge til feilmeldinger på nykunde	Alexander A Jørgensen	Alexander A Jørgensen	↑	FERDIG	Utført	21/apr/21	12/mai/21	
	KUN-157	Backendfunksjon som returnerer alle leverandører (navn og id)	Timian	Alexander A Jørgensen	↑	FERDIG	Utført	21/apr/21	28/apr/21	
	KUN-156	kunde/leverandør, knappene hopper	Alexander A Jørgensen	Alexander A Jørgensen	↑	FERDIG	Utført	21/apr/21	05/mai/21	
	KUN-155	Venter 0,5 sekunder på leverandør/kundeside.	*Ikke tildelt*	Alexander A Jørgensen	↑	FERDIG	Utført	21/apr/21	28/apr/21	

T	ID	Sammendrag	Ansvarlig	Innmelder	Pri	Status	Løsning	Laget	Oppdatert	Frist
	KUN-154	Redesign av svarside for mail	Didrik Bjerk	Alexander A Jørgensen		FERDIG	Utført	21/apr/21	12/mai/21	
	KUN-153	Sende mail til kunder fra en leverandør	*Ikke tildelt*	Alexander A Jørgensen		GJØREMÅL	*Uløst*	21/apr/21	21/apr/21	
	KUN-152	Implementere responsivt design	Kristoffer Haugen	Alexander A Jørgensen		EVALUATION	*Uløst*	21/apr/21	20/mai/21	
	KUN-151	legge til leverandør på kunde	Alexander A Jørgensen	Alexander A Jørgensen		FERDIG	Utført	21/apr/21	28/apr/21	
	KUN-149	Sende mail til flere kunder/leverandører samtidig	Alexander A Jørgensen	Alexander A Jørgensen		FERDIG	Utført	21/apr/21	27/apr/21	
	KUN-148	KUN-138 Bruke nøkkelord, ord i mailen	*Ikke tildelt*	Timian		FERDIG	Utført	14/apr/21	12/mai/21	
	KUN-147	KUN-114 Frontend	Kristoffer Haugen	Timian		FERDIG	Utført	14/apr/21	12/mai/21	
	KUN-146	KUN-114 Backend	Timian	Timian		FERDIG	Utført	14/apr/21	12/mai/21	
	KUN-145	KUN-40 Implementere i frontend	*Ikke tildelt*	Timian		GJØREMÅL	*Uløst*	14/apr/21	14/apr/21	
	KUN-144	KUN-40 Design i FIGMA	*Ikke tildelt*	Timian		GJØREMÅL	*Uløst*	14/apr/21	14/apr/21	
	KUN-143	KUN-40 Implementasjon i backend	Didrik Bjerk	Timian		GJØREMÅL	*Uløst*	14/apr/21	14/apr/21	
	KUN-142	KUN-40 Research	Didrik Bjerk	Timian		GJØREMÅL	*Uløst*	14/apr/21	14/apr/21	
	KUN-141	Internasjonalisering funksjonalitet	Kristoffer Haugen	Alexander A Jørgensen		FERDIG	Utført	07/apr/21	14/apr/21	
	KUN-140	Endre språk: legge til funksjonalitet	Kristoffer Haugen	Alexander A Jørgensen		FERDIG	Utført	07/apr/21	13/apr/21	
	KUN-139	Endre rekkefølge på vis mail (backend funksjon)	Didrik Bjerk	Alexander A Jørgensen		FERDIG	Utført	07/apr/21	08/apr/21	
	KUN-138	Søkefelt i vis mail	Kristoffer Haugen	Alexander A Jørgensen		FERDIG	Utført	07/apr/21	06/mai/21	
	KUN-137	Skaffe SSL-sertifikat for nettsiden	Didrik Bjerk	Alexander A Jørgensen		FERDIG	Utført	07/apr/21	08/apr/21	
	KUN-136	Endre URL til flyt.cloud	Didrik Bjerk	Alexander A Jørgensen		FERDIG	Utført	07/apr/21	09/apr/21	
	KUN-135	Feilmelding når applikasjonen ikke kan hente data	Kristoffer Haugen	Alexander A Jørgensen		FERDIG	Utført	07/apr/21	21/apr/21	
	KUN-134	Feilmelding i frontend når ansatt ikke har tilgang på en kunde/leverandør	Kristoffer Haugen	Alexander A Jørgensen		FERDIG	Utført	07/apr/21	21/apr/21	

T	ID	Sammendrag	Ansvarlig	Innmelder	Pri	Status	Løsning	Laget	Oppdatert	Frist
☑	KUN-133	Velge farger frontend	*Ikke tildelt*	Alexander A Jørgensen	↑	FERDIG	Utført	07/apr/21	21/apr/21	
☑	KUN-132	Redesign av siden	Alexander A Jørgensen	Alexander A Jørgensen	↑	FERDIG	Utført	07/apr/21	05/mai/21	
☑	KUN-131	Trykke på leverandør på kundeside for å bli tatt til den siden	Kristoffer Haugen	Alexander A Jørgensen	↑	FERDIG	Utført	26/mar/21	08/apr/21	
☑	KUN-130	Modifisere GetAllEmployee til å også vise navn til kundene	Timian	Timian	↑	FERDIG	Utført	25/mar/21	07/apr/21	
☑	KUN-129	Legge til testbrukere (kunde)	Timian	Alexander A Jørgensen	↑	FERDIG	Utført	24/mar/21	14/apr/21	
☑	KUN-128	Kunder får egen profil når de logger på (backend)	Timian	Alexander A Jørgensen	↑	FERDIG	Utført	24/mar/21	14/apr/21	
☑	KUN-127	Legge til testbrukere (ansatte)	Timian	Alexander A Jørgensen	↑	FERDIG	Utført	24/mar/21	07/apr/21	
☑	KUN-126	Undersøke JSdoc to markdown ?	Timian	Alexander A Jørgensen	↑	FERDIG	Utført	24/mar/21	07/apr/21	
☑	KUN-125	Lage frontend funksjon for sletting av leverandører	Timian	Alexander A Jørgensen	↑	FERDIG	Utført	24/mar/21	07/apr/21	
☑	KUN-124	(backend) sende med mail til mottaker i SendMailCustomer	Didrik Bjerk	Alexander A Jørgensen	↑	FERDIG	Utført	24/mar/21	26/mar/21	
☑	KUN-123	Endre antall mail som blir lastet inn (frontend)	*Ikke tildelt*	Alexander A Jørgensen	↑	GJØREMÅL	*Uløst*	24/mar/21	07/apr/21	
☑	KUN-122	Endre antall mail som blir hentet fra backend (backend-funksjon)	*Ikke tildelt*	Alexander A Jørgensen	↑	GJØREMÅL	*Uløst*	24/mar/21	07/apr/21	
☑	KUN-121	Lag ny backend-funksjon kundetype	Didrik Bjerk	Alexander A Jørgensen	↑	FERDIG	Utført	24/mar/21	01/mai/21	
☑	KUN-120	Bekreftelses-"vindu" for lesebekreftelse av mail	Kristoffer Haugen	Alexander A Jørgensen	↑	FERDIG	Utført	24/mar/21	05/apr/21	
☑	KUN-119	Endre backend-funksjoner så de kan redigere	Didrik Bjerk	Alexander A Jørgensen	↑	FERDIG	Utført	24/mar/21	26/mar/21	
☑	KUN-118	Skrive kapittelet "Konklusjon"	*Ikke tildelt*	Alexander A Jørgensen	↑	GJØREMÅL	*Uløst*	24/mar/21	24/mar/21	
☑	KUN-117	Skrive kapittelet "Diskusjon"	*Ikke tildelt*	Alexander A Jørgensen	↑	GJØREMÅL	*Uløst*	24/mar/21	24/mar/21	
☑	KUN-116	Skrive kapittelet "Tester og kvalitetssikring"	*Ikke tildelt*	Alexander A Jørgensen	↑	FERDIG	Utført	24/mar/21	12/mai/21	
☑	KUN-115	Skrive kapittelet "Deployment"	Kristoffer Haugen	Alexander A Jørgensen	↑	FERDIG	Utført	24/mar/21	05/mai/21	

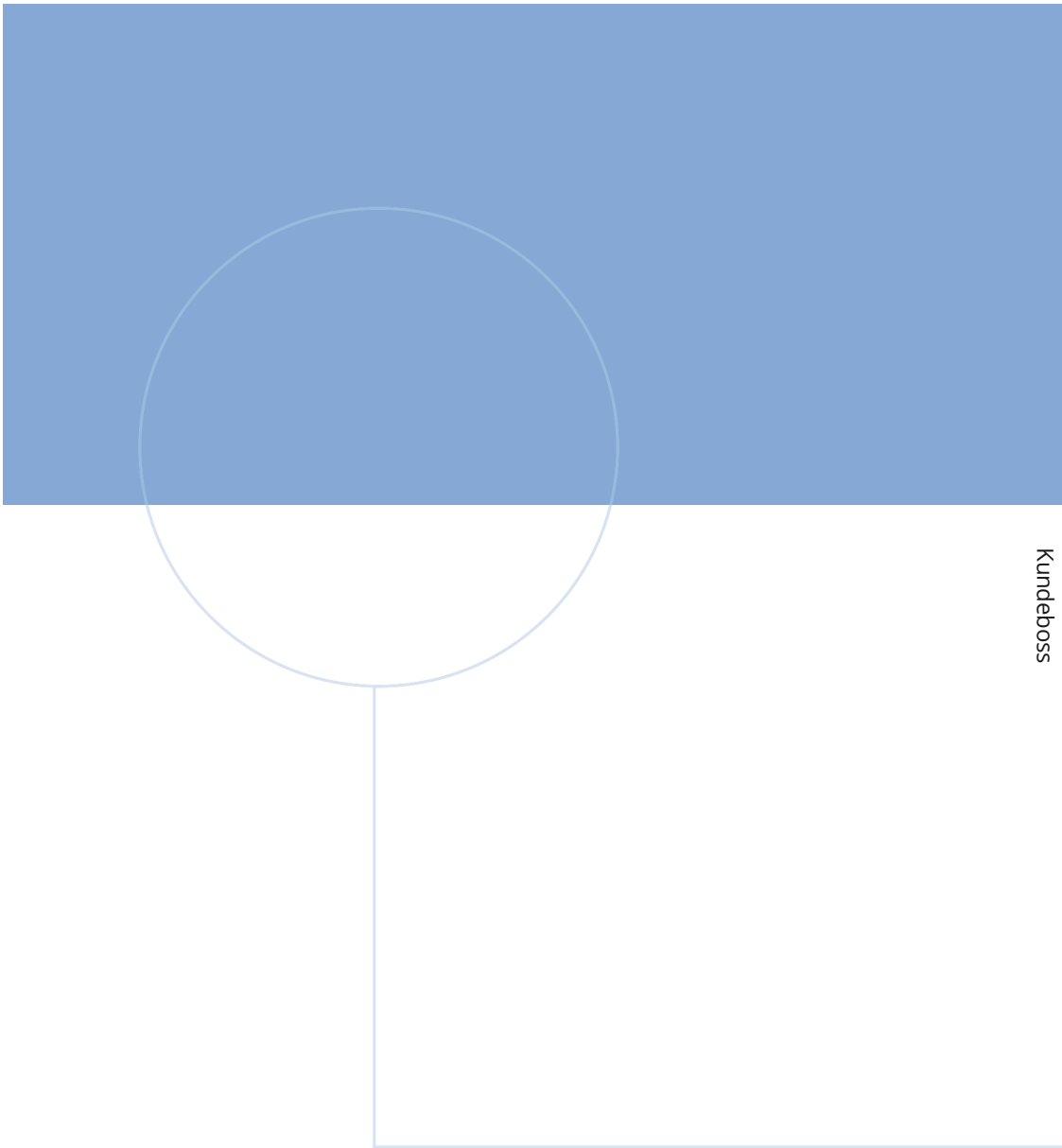
T	ID	Sammendrag	Ansvarlig	Innmelder	Pri	Status	Løsning	Laget	Oppdatert	Frist
	KUN-114	Skrive kapittelet "Implementasjon"	*Ikke tildelt*	Alexander A Jørgensen		FERDIG	Utført	24/mar/21	12/mai/21	
	KUN-113	Skrive kapittelet "System design"	Didrik Bjerk	Alexander A Jørgensen		FERDIG	Utført	24/mar/21	05/mai/21	
	KUN-112	Skrive kapittelet "Utviklingsprosess"	Timian	Alexander A Jørgensen		FERDIG	Utført	24/mar/21	14/apr/21	
	KUN-111	Skrive kapittelet "Kravspesifikasjon"	Kristoffer Haugen	Alexander A Jørgensen		FERDIG	Utført	24/mar/21	07/apr/21	
	KUN-110	Skrive kapittelet "Introduksjon"	Kristoffer Haugen	Alexander A Jørgensen		FERDIG	Utført	24/mar/21	07/apr/21	
	KUN-109	Skrive kapittelet "Brukergrensesnitt og design"	Alexander A Jørgensen	Alexander A Jørgensen		FERDIG	Utført	24/mar/21	08/apr/21	
	KUN-108	Svare på mail	Didrik Bjerk	Timian		FERDIG	Utført	17/mar/21	17/mar/21	
	KUN-107	Kommentere mail	Didrik Bjerk	Timian		FERDIG	Utført	17/mar/21	17/mar/21	
	KUN-105	Leverandørside (FIGMA)	*Ikke tildelt*	Timian		FERDIG	Utført	17/mar/21	24/mar/21	
	KUN-104	Slette ansatte	Timian	Timian		FERDIG	Utført	17/mar/21	24/mar/21	
	KUN-103	Lage API dokumentasjon	Didrik Bjerk	Timian		FERDIG	Utført	17/mar/21	07/apr/21	
	KUN-102	Gjøre kundeside/leverandørside til "samme" komponent	Alexander A Jørgensen	Alexander A Jørgensen		FERDIG	Utført	12/mar/21	26/mar/21	
	KUN-101	Gjøre dashboard knapper trykkbare	Alexander A Jørgensen	Alexander A Jørgensen		FERDIG	Utført	12/mar/21	26/mar/21	
	KUN-100	KUN-45 Redirect etter login	Kristoffer Haugen	Timian		FERDIG	Utført	10/mar/21	22/mar/21	
	KUN-99	Implementere funksjon (leverandørside)	Didrik Bjerk	Timian		FERDIG	Utført	10/mar/21	26/mar/21	
	KUN-98	Implementere funksjon (ny kunde)	Timian	Timian		FERDIG	Utført	10/mar/21	17/mar/21	
	KUN-97	Motta mail	Didrik Bjerk	Timian		FERDIG	Utført	10/mar/21	16/mar/21	
	KUN-95	Implementere funksjonalitet (leverandørside)	Kristoffer Haugen	Timian		FERDIG	Utført	10/mar/21	16/mar/21	
	KUN-94	Implementere side (leverandørside)	Kristoffer Haugen	Timian		FERDIG	Utført	10/mar/21	17/mar/21	
	KUN-93	Design av side for å legge til nye leverandører (FIGMA)	Kristoffer Haugen	Timian		FERDIG	Utført	10/mar/21	16/mar/21	
	KUN-92	Implementere funksjonalitet (ny kunde)	Alexander A Jørgensen	Timian		FERDIG	Utført	10/mar/21	21/apr/21	
	KUN-91	Implementere side (ny kunde)	Alexander A Jørgensen	Timian		FERDIG	Utført	10/mar/21	24/mar/21	

T	ID	Sammendrag	Ansvarlig	Innmelder	Pri	Status	Løsning	Laget	Oppdatert	Frist
	KUN-90	Design av side for å legge til nye kunder (FIGMA)	Alexander A Jørgensen	Timian		FERDIG	Utført	10/mar/21	24/mar/21	
	KUN-89	KUN-31 Research på url	Didrik Bjerk	Timian		FERDIG	Utført	03/mar/21	10/mar/21	
	KUN-88	KUN-28 Funksjon for å sende mail	Didrik Bjerk	Timian		FERDIG	Utført	03/mar/21	09/mar/21	
	KUN-87	KUN-28 Implementere side	Alexander A Jørgensen	Timian		FERDIG	Utført	03/mar/21	21/apr/21	
	KUN-86	KUN-28 Designe side (FIGMA)	Kristoffer Haugen	Timian		FERDIG	Utført	03/mar/21	05/mar/21	
	KUN-85	KUN-41 Implementere side med beskjed til bruker om å henvende seg til sys admin	Alexander A Jørgensen	Timian		FERDIG	Utført	03/mar/21	10/mar/21	
	KUN-84	KUN-31 Hente alt om en kunde	Timian	Timian		FERDIG	Utført	03/mar/21	08/mar/21	
	KUN-83	Endre språk	*Ikke tildelt*	Alexander A Jørgensen		FERDIG	Utført	28/feb/21	13/apr/21	
	KUN-81	Kommentere kode	*Ikke tildelt*	Timian		FERDIG	Utført	24/feb/21	17/mar/21	
	KUN-80	KUN-78 Implementere tester for denne sprinten	*Ikke tildelt*	Timian		FERDIG	Utført	24/feb/21	28/apr/21	
	KUN-79	KUN-78 Test research	*Ikke tildelt*	Timian		FERDIG	Utført	24/feb/21	21/apr/21	
	KUN-78	Teste funksjonalitet	*Ikke tildelt*	Timian		FERDIG	Utført	24/feb/21	28/apr/21	
	KUN-77	Legge til søkefunksjon i rettighetside	Kristoffer Haugen	Timian		FERDIG	Utført	24/feb/21	20/apr/21	
	KUN-72	Definere kondestandard, evt. en addon	*Ikke tildelt*	Timian		FERDIG	Utført	24/feb/21	10/mar/21	
	KUN-71	KUN-34 Display søkeresultatet	Kristoffer Haugen	Timian		FERDIG	Utført	24/feb/21	05/apr/21	
	KUN-70	KUN-69 Velge fargesett	*Ikke tildelt*	Timian		FERDIG	Utført	24/feb/21	03/mar/21	
	KUN-69	Logo design	Alexander A Jørgensen	Timian		FERDIG	Utført	24/feb/21	03/mar/21	
	KUN-68	KUN-34 Hente søkeresultatet fra database	Didrik Bjerk	Timian		FERDIG	Utført	24/feb/21	02/mar/21	
	KUN-67	KUN-34 Søke i database	Didrik Bjerk	Timian		FERDIG	Utført	24/feb/21	02/mar/21	
	KUN-66	KUN-34 Research og diskusjon	Timian	Timian		FERDIG	Utført	24/feb/21	03/mar/21	
	KUN-65	KUN-34 GUI søkeboks	Alexander A Jørgensen	Timian		FERDIG	Utført	24/feb/21	07/mar/21	
	KUN-63	KUN-41 Research og diskusjon om roller (møte med JF)	*Ikke tildelt*	Timian		FERDIG	Utført	24/feb/21	03/mar/21	
	KUN-61	KUN-31 Begrense tilgang for kunde	Kristoffer Haugen	Timian		FERDIG	Utført	24/feb/21	24/mar/21	

T	ID	Sammendrag	Ansvarlig	Innmelder	Pri	Status	Løsning	Laget	Oppdatert	Frist
	KUN-60	KUN-31 Funksjon for å hente kunder	Timian	Timian	↑	FERDIG	Utført	24/feb/21	03/mar/21	
	KUN-59	KUN-31 Funksjon for å legge til kunde	Didrik Bjerk	Timian	↑	FERDIG	Utført	24/feb/21	28/feb/21	
	KUN-58	KUN-31 Implementasjon av side	Alexander A Jørgensen	Timian	↑	FERDIG	Utført	24/feb/21	10/mar/21	
	KUN-57	KUN-31 Design av side (FIGMA)	Alexander A Jørgensen	Timian	↑	FERDIG	Utført	24/feb/21	02/mar/21	
	KUN-56	Research på mail	Didrik Bjerk	Timian	↑	FERDIG	Utført	24/feb/21	16/mar/21	
	KUN-55	Opprette rutine for møter og andre aktiviteter	Alexander A Jørgensen	Timian	↑	FERDIG	Utført	24/feb/21	03/mar/21	
	KUN-54	Lese en hel rapport (og notere)	*Ikke tildelt*	Timian	↑	FERDIG	Utført	24/feb/21	17/mar/21	
	KUN-53	Legge til beskrivelse på Use case	Alexander A Jørgensen	Timian	↑	FERDIG	Utført	24/feb/21	03/mar/21	
	KUN-52	KUN-18 Autentisering i backend	Timian	Alexander A Jørgensen	↑	FERDIG	Utført	17/feb/21	24/feb/21	
	KUN-51	KUN-30 Legge inn informasjon i DB	Didrik Bjerk	Alexander A Jørgensen	↑	FERDIG	Utført	17/feb/21	24/feb/21	
	KUN-50	KUN-30 Implementasjon av siden	Alexander A Jørgensen	Alexander A Jørgensen	↑	FERDIG	Utført	17/feb/21	24/feb/21	
	KUN-49	KUN-30 Enkel design av side	Kristoffer Haugen	Alexander A Jørgensen	↑	FERDIG	Utført	17/feb/21	17/feb/21	
	KUN-48	KUN-30 Sende tokens fra front til back	Timian	Alexander A Jørgensen	↑	FERDIG	Utført	17/feb/21	24/feb/21	
	KUN-47	KUN-30 Research: hvordan beskytte backend	Didrik Bjerk	Alexander A Jørgensen	↑	FERDIG	Utført	17/feb/21	19/feb/21	
	KUN-46	KUN-18 Display informasjon i dashboard	Alexander A Jørgensen	Alexander A Jørgensen	↑	FERDIG	Utført	17/feb/21	26/feb/21	
	KUN-45	Authorisering	*Ikke tildelt*	Alexander A Jørgensen	↑	FERDIG	Utført	17/feb/21	22/mar/21	
	KUN-44	KUN-18 Implementere websiden	Kristoffer Haugen	Alexander A Jørgensen	↑	FERDIG	Utført	17/feb/21	24/feb/21	
	KUN-43	KUN-18 Enkel design av startside	Alexander A Jørgensen	Alexander A Jørgensen	↑	FERDIG	Utført	17/feb/21	17/feb/21	
	KUN-41	Første pålogging	*Ikke tildelt*	Alexander A Jørgensen	↑	FERDIG	Utført	16/feb/21	10/mar/21	

T	ID	Sammendrag	Ansvarlig	Innmelder	Pri	Status	Løsning	Laget	Oppdatert	Frist
	KUN-40	Oppdatere database	*Ikke tildelt*	Alexander A Jørgensen		GJØREMÅL	*Uløst*	16/feb/21	21/apr/21	
	KUN-37	Legge til nye ansatt	*Ikke tildelt*	Alexander A Jørgensen		FERDIG	Utført	16/feb/21	10/mar/21	
	KUN-36	Slette Leverandør	Timian	Alexander A Jørgensen		FERDIG	Utført	16/feb/21	07/apr/21	
	KUN-35	Slette kunde	Timian	Alexander A Jørgensen		FERDIG	Utført	16/feb/21	24/mar/21	
	KUN-34	Søke etter tag(s)	Kristoffer Haugen	Alexander A Jørgensen		FERDIG	Utført	16/feb/21	05/apr/21	
	KUN-33	Se leverandørside	Alexander A Jørgensen	Alexander A Jørgensen		FERDIG	Utført	16/feb/21	06/apr/21	
	KUN-31	Se kundeside	*Ikke tildelt*	Alexander A Jørgensen		FERDIG	Utført	16/feb/21	24/mar/21	
	KUN-30	Autentisering / login	*Ikke tildelt*	Alexander A Jørgensen		FERDIG	Utført	16/feb/21	24/feb/21	
	KUN-29	Send mail til flere	*Ikke tildelt*	Alexander A Jørgensen		FERDIG	Utført	16/feb/21	10/mar/21	
	KUN-28	Sende ny mail	Alexander A Jørgensen	Alexander A Jørgensen		FERDIG	Utført	16/feb/21	18/apr/21	
	KUN-27	Lese mail til/fra kunde	Alexander A Jørgensen	Alexander A Jørgensen		FERDIG	Utført	16/feb/21	07/apr/21	
	KUN-23	Legge til ny leverandør (og redigere leverandør)	*Ikke tildelt*	Alexander A Jørgensen		FERDIG	Utført	16/feb/21	07/apr/21	
	KUN-22	Legge til ny kunde (og redigere kunde)	*Ikke tildelt*	Alexander A Jørgensen		FERDIG	Utført	16/feb/21	07/apr/21	
	KUN-21	Individuelt arbeid / research	*Ikke tildelt*	Alexander A Jørgensen		FERDIG	Utført	10/feb/21	17/feb/21	
	KUN-20	Roller i AD -> Roller i databasen	Timian	Alexander A Jørgensen		FERDIG	Utført	10/feb/21	24/feb/21	
	KUN-18	Se dashboard	*Ikke tildelt*	Alexander A Jørgensen		FERDIG	Utført	10/feb/21	03/mar/21	
	KUN-17	Velge navn	Alexander A Jørgensen	Alexander A Jørgensen		FERDIG	Utført	10/feb/21	02/mar/21	
	KUN-16	(Bli ferdig med) database modellering	Didrik Bjerk	Alexander A Jørgensen		FERDIG	Utført	10/feb/21	17/feb/21	

T	ID	Sammendrag	Ansvarlig	Innmelder	Pri	Status	Løsning	Laget	Oppdatert	Frist
☑	KUN-15	Utkast til design (figma)	Kristoffer Haugen	Alexander A Jørgensen	↑	FERDIG	Utført	10/feb/21	17/feb/21	
☑	KUN-14	Koble frontend og backend	Didrik Bjerk	Alexander A Jørgensen	↑	FERDIG	Utført	10/feb/21	24/feb/21	
☑	KUN-13	Legge inn use caser i backlog	Alexander A Jørgensen	Alexander A Jørgensen	↑	FERDIG	Utført	10/feb/21	17/feb/21	
☑	KUN-12	Godkjenne use caser	Alexander A Jørgensen	Alexander A Jørgensen	↑	FERDIG	Utført	10/feb/21	17/feb/21	
☑	KUN-11	Koble sammen server, database, nettside, autentisering	Alexander A Jørgensen	Alexander A Jørgensen	↑	FERDIG	Utført	03/feb/21	17/feb/21	
☑	KUN-10	Jobbe med design av nettsiden	Kristoffer Haugen	Alexander A Jørgensen	↑	FERDIG	Utført	03/feb/21	17/feb/21	
☑	KUN-9	Skrive ferdig omfang og noe kravspesifikasjon	Kristoffer Haugen	Alexander A Jørgensen	↑	FERDIG	Utført	03/feb/21	16/feb/21	
☑	KUN-8	Lage enkel nettside	Kristoffer Haugen	Alexander A Jørgensen	↑	FERDIG	Utført	03/feb/21	10/feb/21	
☑	KUN-7	Sette opp server og database	Didrik Bjerk	Alexander A Jørgensen	↑	FERDIG	Utført	03/feb/21	10/feb/21	
☑	KUN-5	Skrive sikkerhetskrav og operasjonelle krav	Alexander A Jørgensen	Alexander A Jørgensen	↑	FERDIG	Utført	03/feb/21	16/feb/21	
☑	KUN-4	Lage usecaser	Alexander A Jørgensen	Alexander A Jørgensen	↑	FERDIG	Utført	03/feb/21	11/feb/21	
☑	KUN-3	Sette opp autentifisering	Timian	Alexander A Jørgensen	↑	FERDIG	Utført	03/feb/21	09/feb/21	



Kundeboss

NTNU
Kunnskap for en bedre verden