UiA Universitetet i Agder

NimBook

Plattform for booking av møterom i Microsoft-baserte bedrifter.

Nimbus
Nikolai Holmen Dyb, Mats Eide Skjærvik, Simen Fredrik Brunvand Fredriksen, Nils Fredrik Iselvmo Bjerk, Marius Granum Norli, Vebjørn Nordby, Emil Morønning Ruud.

Veileder
Devendra Bahadur Thapa

Universitetet i Agder, 2019
Fakultetet for Samfunnsvitenskap
Institutt for Informasjonssystemer

Forord

Takk til Merethe Sjøberg, produkteier hos EVRY, for nøye oppfølging, gode råd, motivasjon og evaluering av prosjektet. Vi takker for at du har holdt oss på riktig sti når fokuset har vært på feil plass.

Takk til Espen Limi for ærlige og kritiske tilbakemeldinger, samt god veiledning når vi har stått ved veiskiller i prosjektet.

Takk til Christian Moen for mye teknisk hjelp, og gode innspill gjennom planlegging, og utviklingen av systemet vårt.

Vi ønsker å takke EVRY og deres ansatte for et flott arbeidslokale og et trivelig arbeidsmiljø.

Takk til vår veileder Devinder Bahadur Thapa for gode tilbakemeldinger og innspill under prosjektet.

Takk til Hallgeir Nilsen for å være tilgjengelig for spørsmål og svar gjennom hele prosjektperioden.

Sammendrag

Som en siste innlevering av bacheloroppgave ved Universitetet i Agder har vi i siden Januar 2019 arbeidet med et bachelorprosjekt i samarbeid med EVRY Kristiansand og Universitetet i Agder. Vårt prosjekt har vært å utvikle et informasjonssystem for møteroms-reservasjon med mulighet for verifisering av oppmøte, da dette var noe arbeidsgiver uttrykket behov for. Vi begynte en intervjurunde for å danne oss et bilde av hva arbeidsgiver kunne tenke seg, og utarbeidet utifra fra dette et sett med brukerhistorier som igjen ble brutt ned i mindre deler for å gjøre det mer håndterbart under utviklingen.

I dette prosjektet har vi valgt å bruke arbeidsmetoden Scrum. Scrum ble et naturlig valg siden oppdragsgiver og vi har erfaring med metoden fra før av. Vi følte metoden passet bra til vårt prosjekt. Vi har hele veien arbeidet agilt og dette betyr at krav og prioriteringer har endret seg underveis, noe vi har måtte ta høyde for. For å sikre kvaliteten i produkte satte vi tidlig opp krav og regler for måten vi skulle utvikle prosjektet. Her var kodekontroll og involvering av produkteier viktige tiltak for å opprettholde kvalitet.

Gruppen har gjennom prosjektet fått læringsutbytte med å jobbe i et gruppebasert agilt prosjekt. Vi har lært oss nye programmeringsspråk innen både frontend og backend samt lær oss å bruke mange forskjellige verktøy til både utvikling og gruppestyring.

Resultatet for prosjektet ble en Microsoft-basert webapplikasjon som gjør reservasjon av møterom enkelt, og løser hverdagsproblemer i bedrifter der møterom står tomme. Vi har implementert alle de høyest prioriterte brukerhistoriene produkteier satte krav for, og mener sluttproduktet har preg av god kvalitet.

Innholdsfortegnelse

Figurliste

1. Introduksjon

Dette dokumentet beskriver arbeidsprosessen i vårt bachelorprosjekt ved Universitetet i Agder (UiA), hvor vi har hatt EVRY som oppdragsgiver. Dokumentet vil fungere som en del av innleveringen i faget IS-304, i form av dokumentasjonen til et systemutviklingsprosjekt.

EVRY er Norges største IT-selskap, med ansvar for omtrent en tredel av alle IT-tjenesteleveranser både i privat og offentlig sektor. Hovedvekten av selskapet ligger innenfor norden, spesifikt i Norge og Sverige. I tillegg til har EVRY også kontorer i USA, Ukraina og India (EVRY, 2019). Basert på dette ønsket vi EVRY som oppdragsgiver fordi vi så for oss en arbeidsgiver med høye krav til kvalitet, som kan bidra til at dette i høyere grad blir et utfordrende og lærerikt prosjekt. Våre kontaktpersoner i EVRY er Merethe Sjøberg, som er produkteier, og Espen Limi, som vil være ansvarlig for veiledning på den tekniske delen.

Formålet med prosjektet er å utvikle et system for møteroms-reservasjon med verifisering av oppmøte. Vi har gitt systemet navnet “NimBook”. Det er et rimelig og skalerbart system som enkelt kan brukes i Microsoft-baserte organisasjoner.

Ideen til prosjektet kom frem i samtaler med EVRY, hvor det kom frem at de hadde utfordringer ved bruk av sitt nåværende system i form av at rom som ble booket ikke alltid ble brukt. Vi ønsket derfor å utvikle en funksjon i det nye systemet som verifiserer oppmøte på rommene som er booket, og som evt frigjør rommet til andre dersom oppmøte ikke verifiseres i den perioden som rommet er reservert.

2. Produktet

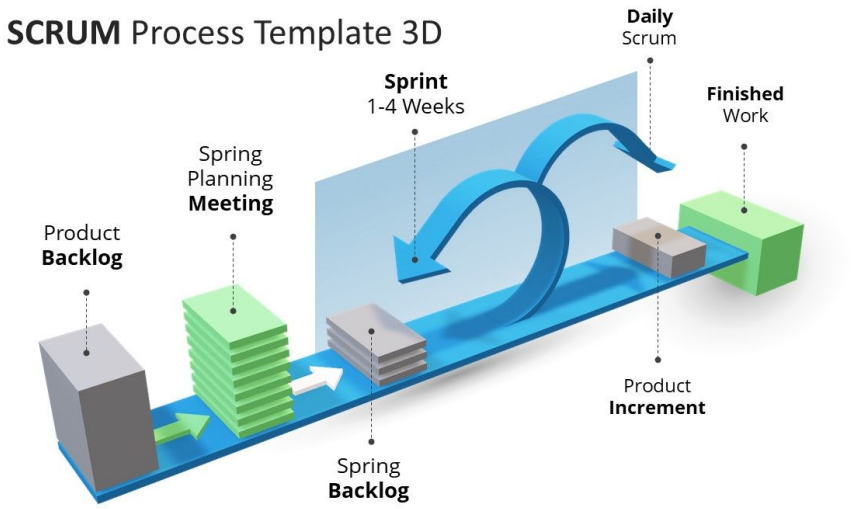
NimBook er et nettbasert reservasjonssystem for møterom. Systemet er bygget for å integrere med Microsoft-produkter, og kan tas i bruk av hvilken som helst Microsoft-basert bedrift. Det er utviklet for å løse utfordringer som for eksempelvis overlapping av reservasjoner og ubrukte rom. Lenke til film: (https://youtu.be/9iX-1t8Hr68)

3. Sentrale avgjørelser i prosjektet

Dette kapittelet beskriver hvilke valg vi har gjort i løpet av prosjektet, og en begrunnelse samt konsekvenser av disse valgene.

3.1 Metode og prosjektstyring

På grunn av usikkerheten i prosjektet, og fordi det var et systemutviklingsprosjekt ønsket vi å bruke en agil metodikk som gir rom for endringer underveis i prosjektet. Valget falt på Scrum siden dette er en metodikk vi er kjent med fra før, samt en metodikk EVRY har god kjennskap til. Vi kan derfor dra nytte av deres kompetanse til å bruke Scrum på best mulig måte. Muligheten for endringer underveis i utviklingen gjør at både kunde og utviklere har påvirkningskraft i utviklingsløpet. (Mountain Goat Software, 2019)



Figur 1 - Scrum Process

3.1.1 Sprint

Vi bestemte oss for å ha sprinter på 2 uker i begynnelsen av prosjektet. I begynnelsen av hver sprint avholdt vi planleggingsmøter hvor vi gikk gjennom backloggen og bestemte hvilke brukerhistorier vi skulle arbeide med i sprinten.

3.1.2 Standup

Hver dag hadde vi standup hvor vi snakket om hva vi har gjorde, utfordringer og hva vi skulle gjøre. I løpet av en sprint ser vi at prosjektet har utvikling og at prosessen går videre.

3.1.3 Retrospekt

Når vi avsluttet en sprint oppsummerte vi i form av en retrospekt og gjorde oss klare til en ny sprint. Etter hver sprint fikk vi muligheten til å se tilbake på hvordan arbeidsprosessen har vært, lære av feilene vi har gjort og gjøre endringer, samtidig som vi tok med oss videre det som fungerte godt. Muligheten til å gjøre endringer

underveis, legge til og ta med uferdig arbeid i backlogen sørget for at vi fikk fremgang i prosjektet og jobbet mot et ferdig produkt.

3.1.4 Endringer

Som en gruppe på 7 personer så vi behovet for gode hjelpemidler for å fordele oppgaver og arbeide på en effektiv måte. Gjennom prosjektet har vi endret og tilpasset arbeidsmetoden slik at det skulle fungere best mulig for dette prosjektet og oss som gruppe. Vi har hele tiden jobbet etter Scrum og tilpasset etter det. Den største endringen vi gjorde underveis var å endre lengden på sprintene fra to uker til én uke. Denne avgjørelsen ble tatt fordi vi så at det var vanskelig å planlegge og sette realistiske mål for to uker om gangen, noe som gjorde at vi slet med å fullføre målene vi satte for hver sprint. På grunn av dette gjennomførte vi planlegging hver mandag, noe som gjorde det lettere for oss å vurdere arbeidsmengde, prioritet og ansvar.

I starten var planleggingsmøtene korte og vi bestemte kun hvilke funksjoner som måtte fullføres i løpet av sprinten, uten å egentlig dele de opp i oppgaver. Dette endret vi etterhvert til møter som varte i opptil to timer. Vi delte opp hver oppgave i mindre deler og fordelte hvem som hadde ansvar for hver enkelte oppgave. Slik hadde alle full oversikt på hva som skulle gjøres i løpet av sprinten og hvem som hadde ansvar for hva. Dette gjorde at hele gruppen ble mye mer effektive med arbeidet gjennom sprintene. Vi utarbeidet også en kort plan på hva vi skulle gjennomføre neste sprint, og delte ut ansvarsområder. Grunnen til at vi delte ut ansvar var for å følge opp at arbeidsoppgaver ble utført. (Schwaber and Sutherland, 2017, s.16).

I starten av prosjektet satt vi under standup møtene, og hadde ikke noe fast tidspunkt for møtet. Det hendte at vi glemte standup og at møtene ble veldig lange og lite effektive. Løsningen ble at vi hver dag klokken 09:00 reiste oss opp og gjennomførte standup. På denne måten holdt vi møtene korte og konsise, samt sørget for at vi ikke glemte møtet. På standup fikk vi en oversikt over hva hver enkelt jobbet med og hvilke utfordringer vi møtte på. I starten jobbet vi ofte litt for lenge på en oppgave før vi spurte om hjelp, noe som hemmet fremgangen i prosjektet. Løsningen ble at dersom vi hadde jobbet med samme oppgave to dager på rad tok vi dette opp på standup slik at alle var klar over problemet og kunne hjelpe til eller endre oppgaver etter møtet.

3.1.5 Risikovurdering

Vi utarbeidet en risikomatrise som tok for seg sannsynlighet og konsekvenser ved potensielle hendelser. Her la vi inn tiltak for å forhindre og redusere skaden av risikoene. Dette viste seg å bli et nyttig hjelpemiddel for å vise produkteier hvilke

potensielle farer vi sto ovenfor, og nyttig for oss for å vite hvilke tiltak vi eventuelt må iverksette. En iterasjon av risikomatrisen kan sees under vedlegg(Ref vedlegg 1).

3.1.6 Prosjektstyringsverktøy - Azure DevOps

I prosjektet tok vi i bruk Azure DevOps som prosjektstyringsverktøy. DevOps gir tilgang til en rekke verktøy slik som boards, repos og pipelines. Alle disse verktøyene kommuniserer med hverandre, noe som har gjort det lett for oss å ha en god oversikt over arbeidsoppgaver og repositorier gjennom hele prosjektet. Samtidig har Azure pipelines hjulpet oss med testing og distribuering av kode. En av grunnene til at vi valgte Azure var at det er et godt brukt verktøy i EVRY og vi på denne måten kunne benytte oss av deres kompetanse for å få full nytte av det. I tillegg er Azure DevOps åpen for utvidelser slik vi kunne tilpasse verktøyet ut i fra våre behov.

3.2 Verktøy og språk

Vi har benyttet oss av flere forskjellige rammeverk i løpet av prosjektet. Dette har vi valgt å gjøre da det å bruke rammeverk sparer mye tid i forhold til å lage alt fra bunnen av.

3.2.1 React

I utredningen for å finne front-end rammeverk sto det mellom to valg; React og Angular. Disse to var de mest brukte frontend rammeverkene i bransjen da vi stod overfor valget. (Stateofjs, 2018) Ved å veie opp fordeler og ulemper ved rammeverkene kom vi frem til at React var det beste alternativet for oss. Sentrale punkter i denne vurderingen var at React var mer etterspurt i markedet og at EVRY hadde flere konsulenter med erfaring i React som vi kunne dra nytte av ved behov. I en React komponent skriver man i et JSX-format, slik at man kan skrive både JavaScript og HTML. Disse komponentene kan gjenbrukes og hindrer duplikasjon. React har mange tilgjengelige biblioteker som kan benyttes fritt. Dette gjør at man ikke trenger å utvikle noe som allerede finnes, samt at oppdateringer kan gjøres automatisk. React kan endre en enkelt del av siden uten å oppdatere hele siden på nytt, noe som sørger for en sømløs brukeropplevelse, samt raskere respons.

3.2.2 Node.js

Siden React kun er et front-end rammeverk, er det også behov for et sted vi kan lagre og modifisere data. Valget falt på Node.js. Da vi allerede hadde valgt React som frontend, som er et javascript rammeverk, bestemte vi oss for å velge Node som backend slik at det ble mye likhet i koden. Dette var viktig på grunn av at vi primært er bundet av tid.

3.2.3 GraphQL

Til uthenting av data valgte vi GraphQL. GraphQL er et spørrespråk utviklet av Facebook som lar deg hente akkurat de dataene du trenger. Fordelen med dette er at man får raskere og mer optimale spørringer, som forbedrer brukeropplevelsen. GraphQL er også relativt nytt, og nettopp dette så vi som er sjanse til å tilegne oss kunnskap om ny teknologi.

3.2.4 Microsoft Graph API

Opprinnelig hadde vi tenkt å lagre rom, bookinger og brukere selv ved hjelp av en lokal database løsning basert på MongoDB. I løpet av analysen fikk vi god innsikt i hvordan booking av møterom foregikk i EVRY og fant ut at Microsoft Outlook ble brukt i de nåværende rutinene. I løpet av studiet har vi lært at endringer i en bedrift kan være vanskelig. Dersom Nimbook ikke blir godt tatt imot av alle kunne det blitt problematisk å ha to kilder på reservasjonene. Microsoft Graph API gjør det mulig for oss å behandle all data tilknyttet en Microsoft-bruker. Dette innebærer blant annet at bookinger som blir gjort i vårt system også vil være synlige for de som bruker Outlook. EVRY er partner med Microsoft og benytter seg allerede i stor grad av Microsoft-baserte løsninger.

3.2.5 React Native

Vi ønsket å løse verifisering av oppmøte med en mobilapplikasjon slik at brukere fysisk må være ved rommet for å verifisere sitt oppmøte og dermed sikre at reservasjonen består. Valget stod mellom Flutter og React Native. Disse er begge kryssplattform-rammeverk, som betyr at de fungerer for både iOS og Android. Flutter skrives i Dart, mens React Native skrives i JavaScript. Utviklingen av mobilapplikasjonen begynte tidlig i mars og vi hadde allerede har brukt mye tid på å lære oss nye rammeverk og språk. På grunn av prosjektets tidsbegrensning og fordi vi allerede bruker JavaScript både frontend og backend valgte vi å bruke React Native for å unngå å måtte lære oss et nytt språk.

3.3 Kvalitet i prosjektet

Tidlig i prosjektet var det viktig for oss å definere hva vi anser for å være god kvalitet. Hensikten med dette var å gjøre kvalitet målbart, slik at vi kan teste vårt arbeid opp mot satte kriterier. Dette er en sentral del av SMART-metoden for å sette kriterier i et prosjekt ("What is SMART in Project Management?", 2019). Samtidig vil dette gjøre det enklere for andre utviklere som eventuelt skulle videreutvikle produktet etter oss. Kvaliteten har vi gjennom prosjektgjennomførelse og sluttprodukt opprettholdt ved nøye planlegging og god kommunikasjon mellom oss og produkteier. For en

fullstendig oversikt over våre kvalitetskriterier legges det ved et kvalitetsdokument under vedlegg(Vedlegg 2).

3.3.1 Kode

For å oppnå en god kvalitet i prosjektet har vi benyttet oss av en kodestandard for å gjøre koden lesbar og strukturert. I et prosjekt der man samarbeider med andre for å skrive en kodebase kan det være vanskelig å forstå andre sin kode. Derfor ble vi enige om å følge prinsipper fra *Clean Code* av Robert C. Martin. Et av prinsippene vi har fulgt etter beste evne er: "*The name of a variable, function, or class, should answer all the big questions. It should tell you why it exists, what it does, and how it is used. If a name requires a comment, then the name does not reveal its intent."* (Martin, 2009, s.18). Ved at navnene på klasser, funksjoner og variabler er selvforklarende har vi spart tid på at vi ikke trenger å snakke med forfatteren for å forstå hva som skjer i koden.

Prettier

Prettier er en utvidelse i Visual Studio Code som automatisk formaterer koden. Vi satte Prettier til å formatere koden hver gang en fil lagres. Dermed unngikk vi lange linjer med kode, og gjort at hver enkelt fil er lettere å lese da alle følger samme struktur. Et eksempel på hvordan det fungerer kan sees i figur 2.

Input

```
foo(reallyLongArg(), omgSoManyParameters(), IShouldRefactorThis(), isThereSeriouslyAnotherOne());
```

Output

```
foo(
  reallyLongArg(),
  omgSoManyParameters(),
  IShouldRefactorThis(),
  isThereSeriouslyAnotherOne()
);
```

Figur 2 - Prettier

3.3.2 Involvering av produkteier

Dette er det første prosjektet hvor noen har hatt forventninger og krav til vårt arbeid utenom foreleser. Merethe Sjøberg har fungert som produkteier i prosjektet, og gjorde det tidlig klart at det var vår oppgave å holde henne oppdatert og eller si ifra hvis det var noe vi trengte hjelp til. Alt som ble lagt til, endret eller fjernet i systemet, skulle bli diskutert og godkjent av produkteier (Kroenke & Boyle, 2016, s. 648).

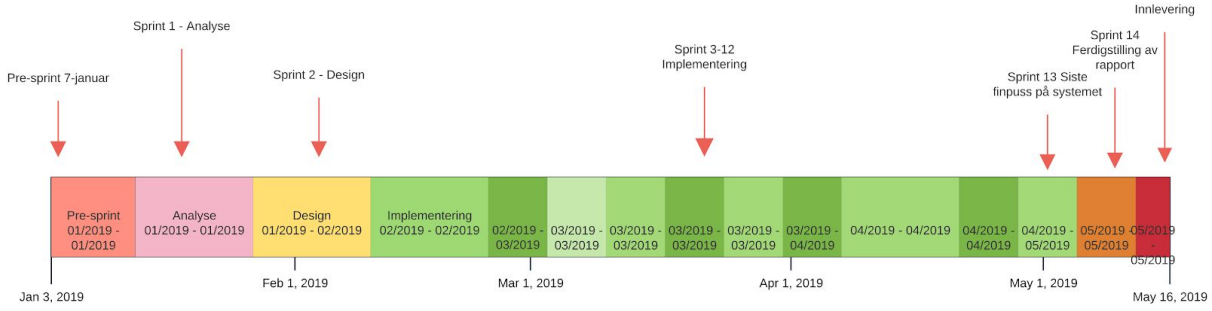
Vi hadde statusmøter i begynnelsen av hver sprint, men produkteier følte ikke dette var tilstrekkelig, og hadde ikke den oversikten hun ønsket over fremgangen i prosjektet. For å rette opp i dette bestemte vi oss sammen med produkteier å gå over til to statusmøter i uken, samt å invitere produkteier til alle planleggingsmøter og retrospekt. Det var svært nyttig for prosjektet med mer involvering av produkteier, fordi vi fikk hyppigere tilbakemeldinger, og endringer ble raskt registrert og utført. Dette medførte at vi unngikk å legge mange timer i arbeid som senere viste seg å være unødvendig.

4. Prosjektgjennomføring

Dette prosjektet har vært mye mer omfattende enn andre prosjekter vi har jobbet med tidligere, og det har derfor vært viktig at vi arbeidet sammen for å kunne levere et system som tilfredsstiller oppdragsgivers krav. For å oppnå dette har vi hatt en tydelig plan og en kultur som promoterer åpenhet og diskusjon som inkluderer alles meninger.

4.1 Planlegging

I begynnelsen av prosjektet brukte vi pre-sprinten til å planlegge hvordan vi skulle fordele tiden og arbeidsmengden i den perioden vi hadde til rådighet. Det viktigste var at vi, sammen med produkteier, bestemte oss for hva systemet skulle være, og utarbeidet kriterier ut ifra dette. Vi delte opp prosjektarbeidet i flere faser. Før vi satte i gang med fasene laget vi en tidslinje på de forskjellige periodene for å få en god oversikt. (Se Figur 3)



Figur 3 - tidslinje

4.2 Analyse

For innhenting av informasjon valgte vi å ha et kvalitativt intervju av ansatte hos EVRY. Vi valgte dette fordi det åpnet opp for en diskusjon mellom respondent og intervjuer, slik at respondenten fikk mulighet til å komme med utdypende svar. Målet med intervjuene var å innhente informasjon om hvordan nåværende system

fungerer, hva som fungerer bra og hva som fungerer dårlig, samt hvilke funksjoner som er ønsket i et nytt system. (Ref vedlegg 3)

Etter informasjonsinnhentingen analyserte vi svarene og utformet brukerhistorier basert på disse. (Ref vedlegg 4) Sammen med produkteier prioriterte vi brukerhistoriene og definerte **M**inimum **V**iable **P**roduct (MVP).Alle brukerhistoriene ble rangert ved bruk av MoSCoW-metoden og godkjent av produkteier. Underveis i prosjektet har vi støtt på utfordringer og endringer, som har gjort det nødvendig å revurdere noen av brukerhistoriene og gjøre endringer. Endringene ble diskutert med produkteier i møter hvor vi argumenterte for endringene vi mente var nødvendige.

4.3 Design

I designfasen benyttet vi informasjon fra analysefasen for å designe systemet, hvordan det skulle bygges opp, se ut og hvilke funksjoner det skulle inneholde.

4.3.1 Wireframe

For å få et utgangspunkt for utseende på systemet skisserte vi ulike design av hvordan det skulle se ut. Ut i fra disse laget vi wireframes som senere ble brukt da vi begynte med frontend utviklingen. Wireframes ga oss nytteverdi i form av at produkteier kunne gi oss tilbakemelding på brukergrensesnittet og komme med innspill på hva som kunne endres. Samtidig ble produkteier trygg på hvordan systemet skulle bli seende ut. Et eksempel på en wireframe kan sees under vedlegg(Ref vedlegg 5).

4.4 Implementasjon

Utviklingen av systemet har vært en sentral del i dette bachelorprosjektet. I og med at vi valgte å programmere i kodespråk som ingen av oss hadde brukt før, måtte vi finne en metode for å lære oss programmeringsspråkene så godt som mulig. For å effektivisere programmeringen i det lange løp, brukte vi de to første ukene av utviklingsfasen på å lære oss de nye kodespråkene gjennom kurs på nettet, spesielt gjennom videoer på Youtube. Under denne læringsfasen kodet vi simple web applikasjoner, med fokus på at applikasjonene skulle være så relevante for prosjektet som mulig slik at de kunne gjenbrukes.

Selv om vi satt av tid til å lære oss programmeringsspråkene i starten, er det begrenset hvor mye kunnskap man tilegner seg etter to uker med læring, noe som gjorde at det allikevel ble litt prøving og feiling. Derfor besluttet vi å samarbeide gjennom hele utviklingsfasen, spesielt parprogrammering var en viktig faktor for god fremgang. Vi hadde ofte intern kodegjennomgang, hvor alle forklarte koden man

hadde jobbet med. Kodegjennomgangene ga hele gruppen god oversikt over koden, samt et bedre læringsutbytte.

4.5 Testing

For å forsikre at all kode som ble skrevet eller endret ikke skulle ødelegge for ønsket funksjonalitet utformet vi enhetstester. Disse enhetstestene ble automatisk kjørt i Azure DevOps for hver gang en endring i koden ble utført. Dette gjorde at vi for hver endring fikk svar innen kort tid om testene ble godkjent, eller om det var en feil i koden, og hvor feilen lå. Hvis tilbakemelding var negativ etter gjennomførte tester ble det umiddelbart gjort til førsteprioritet å rette opp feilen som ble oppdaget. Når vi arbeidet med kode brukte vi en praksis som kalles **C**ontinuous **I**ntegration(CI), som vil si at prosjektet bygges og kjører tester når kode pushes til en felles branch (Guckenheimer, 2017). I tillegg til enhetstester ble det utført funksjonell testing hvor vi testet om ulike funksjoner fungerte, og om det var bugs i systemet.

4.5.1 FAT

Mot slutten av prosjektet gjennomførte vi **F**actory **A**cceptance **T**ester sammen med produkteier. Formålet var å teste akseptansetestene til alle brukerhistoriene som var ferdigstilte, for å avdekke om de utførte sine oppgaver ut fra kriteriene vi hadde satt. Her fikk vi tilbakemelding fra produkteier på brukerhistorier som ikke møtte kravene og tid til å utbedre. FAT er med å sikre at vi leverer et produkt som fungerer i henhold til gitte kriterier. Eksempel på FAT referat i vedlegg(Ref vedlegg 6).

4.6 Tidsstyring

Da dette prosjektet ikke består av noe budsjett i form av kroner og øre, men heller er bundet av tid bestemte vi oss tidlig for at vi måtte ha noen form for styring og oversikt over timebruken. Vi gjorde en kalkyle på forventet arbeidsmengde i faget, og la dette som grunnlag for hvor mye vi minimum måtte arbeide med prosjektet. For å holde oversikt over timene brukte vi Clockify (Clockify, 2019). Clockify lar en person spore sine egne arbeidstimer i et prosjekt. Dette ga oss en oversikt over hvor mange timer hver person har lagt inn i prosjektet, samt hvor mange timer som har blitt brukt totalt.

5. Resultat

Innledningsvis var formålet med prosjektet å utvikle et system for møteroms-reservasjon. Det endelige prosjektet har blitt relativt likt det planlagte produktet, men med noen endringer underveis. Vi har dog implementert alt vi og produkteier så som nødvendig, selv om vi selvfølgelig gjerne skulle ha implementert flere funksjoner. Systemet vi endte opp med har funksjonalitet for innlogging med Microsoft-bruker, reservasjon av møterom, hindrer at reservasjoner går samtidig,

oversikt med status for hvert enkelt rom og muligheten til å verifisere at man har tatt i bruk rommet ved hjelp av QR koder og en mobilapplikasjon slik at andre brukere kan se om man har benyttet seg av reservasjonen. Vi mener systemet bærer preg av god kvalitet på bakgrunn av våre rettningslinjer for kvalitetssikring.

6. Refleksjon

Det å jobbe iterativt er en essensiell del av å jobbe agilt. Det innebærer å gå over funksjonalitet man har implementert flere ganger for å avdekke mangler og gjøre forbedringer, men arbeidsprosessen er også et aspekt av prosjektarbeid som bør vurderes. (Kroenke & Boyle, 2016, s. 395) I denne delen av rapporten ser vi på utfordringene vi har hatt underveis og tiltak vi har gjort for å løse disse.

6.1 Hjemmekontor

Da vi begynte prosjektet i januar hadde vi en enighet om at vi skulle behandle prosjektet som en vanlig jobb, slik at vi Mandag t.o.m Torsdag skulle møtes i EVRY sine kontorer å jobbe fra 08.00-15.00. Hvis noen ønsket å jobbe hjemmefra en dag var det i utgangspunktet greit. Etter en stund merket vi at dette viste seg å være ugunstig for prosjektet. De som arbeidet hjemmefra hadde færre aktive arbeidstimer enn de som møtte opp på kontoret, og det ble vanskeligere å samarbeide når hele gruppen ikke var samlet. Et annet problem med hjemmekontor var når noen meldte arbeid hjemmefra ble det fort en kjedereaksjon, og på det verste møtte kun én person opp på kontoret. Etter å ha innsett hvor stort utslag dette hadde på produktiviteten bestemte vi i fellesskap at eneste gyldige fravær fra kontoret var deltidsjobb, sykdom eller jobbintervjuer.

6.2 Standup

En av komponentene i Scrum er å ha en daglig standup hvor vi forteller resten av gruppen hva vi gjorde i går, utfordringer vi har støtt på og hva vi skal gjøre i dag. Hele møtet tar under 5 minutter og er en hurtig måte å orientere seg på hva alle holder på med. (Kroenke & Boyle, 2016, s. 647) I oppstartsfasen hadde vi ingen faste rammer på når vi hadde dette møtet, og vi tok det når de fleste var på plass. Oppmøtet varierte dog litt fra dag til dag, så møtet ble ofte glemt. Vi diskuterte viktigheten med dette møtet og bestemte oss for at møtet skal gjennomføres hver arbeidsdag kl 09:00. Vi kontraktfestet samtidig en regel om at alle skal være på plass til kl 09.00 Mandag t.o.m Torsdag. Brudd på denne regelen medførte en straff hvor vedkommende måtte kjøpe to butikkøl til gruppens “felleskasse”, og en butikkøl ekstra for hver time som går og personen fremdeles ikke har møtt opp. Dette gjorde oppmøtet betraktelig bedre. (Ref vedlegg 7)

6.3 Estimater

For å ha en god oversikt over arbeidet vårt bestemte vi oss for å prioritere oppgaver ut i fra viktighet. Vi erfarte dog ofte at prioriteringene ikke ble respektert, og at folk gjerne tok det de ønsket å jobbe med i stedet for det vi hadde vurdert som viktigst for prosjektet. Ved å gjennomføre standup-møtene som tidligere nevnt, og generelt ha et bedre samarbeid med god kommunikasjon fikk vi en bedre kontroll på hva forskjellige medlemmer i gruppen arbeider med, og kunne lettere se om noen prioriterte noe annet enn det som burde prioriteres.

6.4 Ledelse

Det å ha en gruppe på syv personer gjør at vi må prioritere mye tid på å fordele arbeid. Dette stjeler en del tid, og det er viktig at det blir gjort rett. Det er viktig å hele tiden ha overblikk over hva hvert enkelt medlem skal jobbe med, men det er samtidig ekstremt vanskelig. Det å ha flere medlemmer i gruppen gir mange flere arbeidstimer, men det fører også til at man må prioritere mange av de arbeidstimene til å jobbe med gruppeledelse. På dette området estimerte vi i starten av prosjektet lite tid til å håndtere nettopp dette, derfor møtte vi utfordringer hvor vi ikke ble ferdig med oppgaver i tide. Her måtte vi finne tiltak for hvordan vi skulle forbedre gruppeledelsen, slik at vi alle jobbet med relevante oppgaver for de forskjellige sprintene. Vi begynte prosjektet med en veldig organisert struktur hvor alle medlemmene hadde satte roller og ansvar, men vi gikk senere bort fra dette da vi så det laget såkalte “siloer”. Vi så dog fort at disse siloene ikke hjalp prosjektet, men heller arbeidet i mot oss (Albrecht, 2019, s. 3). Vi byttet derfor til en flatere struktur uten noen definerte roller for å oppmuntre for samarbeid, samt at hvert medlem skulle ha et større eierskap til hele prosjektet, fordi de faktisk har vært med på å utvikle alle områdene.

Dette fungerte også godt en periode, og vi arbeidet med veldig god effektivitet en stund. Vi arbeidet dog ofte litt i feil retning. Og sammen med oppdragsgiver innså vi at det trengs en lederfigur, selv i en flat struktur. Vi bestemte oss derfor for å utnevne et av medlemmene som en leder i prosjektet, som tok de vanskelige avgjørelsene og hadde et overblikk over prosjektet som en helhet. Dette førte oss tilbake i en bedre retning, og slik holdt vi prosessen gående resten av prosjektet.

7. Uttalelse fra EVRY

Attest – Bachelorprosjekt – Våren 2019

Gjennom arrangementet RefreshIT ble EVRY høsten 2018 kontaktet av flere grupper som ønsket å gjennomføre sin bacheloroppgave hos oss. Vi intervjuet flere grupper, men valget falt til slutt på Nimbus. Veiledere gjennom prosjektet har vært Merethe Sjøberg som veileder på prosjektgjennomføring og Espen Limi som teknisk veileder. Merethe har også hatt rollen som kunde og produkteier i prosjektet.

Studentene sto i utgangspunktet fritt til å velge prosjekt hos oss, men etter litt brainstorming mellom EVRY og Nimbus, så kom vi frem til at gruppen skulle utvikle et møtebookingssystem med mulighet for å verifisere oppmøte på møterom. Dette prosjektet ble presentert av EVRY som et problemområde, da vi har begrenset med møterom fordelt på ganske mange ansatte. Vi opplever til tider at mange rom kan stå som booket i romoversikten i Outlook, uten at rommene faktisk er i bruk, noe som er til stor frustrasjon dersom man har behov for rom.
EVRY ønsket derfor at det på en eller annen måte skulle kunne verifiseres om man faktisk har tatt rommet i bruk, slik at møterommet blir tilgjengelig igjen dersom det ikke er verifisert innen en viss periode. I tillegg var det viktig for oss at det nye systemet kan kommunisere med Outlook, slik at vi fortsatt kan benytte dette verktøyet.
Denne koblingen i tillegg til selve verifiseringen er med på å gjøre oppgaven mer utfordrende enn dersom de skulle laget et helt separat system med egen brukerbase.

Gruppen opplevde utfordringer med å organisere sine syv deltakere. Ved å bruke egen kunnskap og veiledning kom de frem til at de måtte gjøre endringer i metodikken sin. Da dette ble gjort begynte de å se stor fremgang og gruppen endret mentaliteten fra å gå fra å være 7 individer som arbeidet, til å bli ett team som satte pris på hverandres tilbakemeldinger og jobbet bedre sammen og fikk se at de var avhengig av alle som er på teamet. Gruppen har og lært mye teknisk om hvordan benytte MS Office Graph API, moderne autentiseringsmekanismer og React.

Vi i EVRY stiller høye krav til de gruppene som gjennomfører oppgaver hos oss – både til profesjonalitet, lærevilje, rapportering og prosjekteiers involvering i prosjektet. Gruppen har løst dette på en svært bra måte. Prosjektet var utfordrende, da de måtte lære seg nye teknologier, samt at de måtte komme opp med løsninger for både verifisering og autorisering. Gruppen har invitert prosjekteier til all sprint planlegging, samt at vi har hatt ukentlige statusmøter og styringsgruppemøter, i tillegg til at vi har vært involvert i testing. Gruppen har også vært flinke til å ta opp problemer og utfordringer når de støter på dem, i stedet for å vente til neste avtalte møte.

Gjengen har tatt oppgaven svært seriøst og sittet i EVRY sine lokaler 4-5 dager i uken siden begynnelsen av Januar

Kristiansand 15.05.2019

Merethe Sjøberg - Veileder

Espen Limi - Veileder

8. Egenvurdering

Alt i alt er gruppen meget fornøyd med sin innsats i prosjektet. I begynnelsen hadde vi noen antakelser om hvordan prosjektet ville foregå, men det har vist seg å bli helt annerledes enn det vi så for oss. Å være kritisk til egen metode har vært et viktig læringsutbytte og vi har måttet endre måten vi har arbeidet på underveis. Det som har fungert i tidligere prosjekter fungerer ikke nødvendigvis i andre prosjekter med en annen gruppesammensetning og et annet scope. Vi har også har lært å samarbeide som en gruppe og utnytte hverandres kompetanse på en helt annen måte enn tidligere.

Prosjektet har til tider vært utfordrende, men den gode gruppedynamikken har bidratt til at vi har klart å holde motivasjonen oppe. I løpet av prosjektet har vi tilbrakt flere hundre timer sammen, og det kunne blitt tungt hvis vi ikke hadde kommet så godt overens. Dette har også bidratt til å skape et miljø hvor vi kan utfordre hverandres antakelser med fruktbare diskusjoner hvor vi blant annet har kunnet avdekke potensielle risikoer i prosjektet.

Emil Morønning Ruud
I løpet av bachelorprosjektet har jeg fått en del teknisk erfaring når det kommer til bruk av verktøy som React, GraphQL og MongoDB, NodeJS, Azure DevOps og Microsoft Graph API. Når dette er sagt så er nok den viktigste erfaringen mer relatert til selve gruppearbeidet; med fokus på uventede (og forventede) problemer som dukket opp og hvordan vi klarte å løse disse sammen som gruppe. I tillegg har dette prosjektet lært meg viktigheten med god planlegging, sette mål og følge opp disse etterhvert som man kommer videre i prosjektet for å gi alle et bedre/mer oppdatert overblikk. Jeg er veldig takknemlig for at jeg fikk jobbe sammen med så flinke folk (både andre gruppemedlemmer og ansatte i EVRY), og vet at dette vil være en meget positiv erfaring å ha med seg videre i arbeidslivet.

Marius Granum Norli
I dette prosjektet har jeg fungert som scrum master, samt ved siden av dette hatt en mer autoritær rolle. Jeg har også bidratt til prosjektet i form av utvikler. Rollene som scrum master og leder har hatt høyere prioritet, noe som har fungert greit, men det har til tider vært vanskelig å skille mellom scrum master og lederrollen, da disse ikke passer godt sammen. Når det gjelder utviklingen har jeg arbeidet både frontend og backend. Totalt sett har det vært en lærerik opplevelse, og jeg har fått kunnskap på mange forskjellige områder.

Nikolai Holmen Dyb
Gjennom dette prosjektet har jeg tatt med meg mange nye erfaringer og tatt til meg mye ny kunnskap. Jeg har fordypet meg i rammeverk som React og Node.js, samt som jeg har lært å bruke verktøy som MsGraph og Azure devops. Det jeg har bidratt mest med i dette prosjektet har vært å lage forskjellige komponenter i React og hente ut og behandle data fra API. Jeg har utviklet kode for både reservasjon og kansellering av rom i backend og frontend. Jeg har også i tatt meg av litt av den administrative biten av prosjektet som prosjektstyring og kommunikasjon med produkteier. Arbeidet i prosjektet har gitt meg erfaring jeg garantert vil få bruk for i fremtiden.

Nils Fredrik Iselvmo Bjerk
Bachelorprosjektet har vært svært spennende, krevende og lærerikt. I løpet av bachelorprosjektet har jeg fått en større forståelse for systemutvikling og prosjektstyring, noe som jeg kommer til å ta med meg ut i arbeidslivet. Hovedbidraget mitt teknisk i prosjektet har vært på React komponenter til frontend, enhetstester og GraphQL. Noe av det viktigste jeg har lært er samarbeid og hvor viktig det er med risikohåndtering.

Mats Eide Skjærvik
Gjennom arbeidet med bachelorprosjekter har jeg fått nye erfaringer om hvordan det er å jobbe med systemutvikling. Det har det vært spennende å se hvordan vi som gruppe har forbedret oss fra begynnelsen til slutten av dette prosjektet, hvilke valg vi har tatt, og resultat av dette. Prosjektet har vært utfordrende og lærerikt da mye av det vi har arbeidet med har vært nytt for meg og resten av gruppen. Det som jeg i hovedsak har arbeidet med er frontend utvikling i React, og koding av enhetstester til frontend komponenter. Erfaringen jeg har tilegnet meg gjennom arbeidet med mine gruppemedlemmer, og kunnskapen EVRY har gitt meg under dette prosjektet er noe jeg kommer til å ta med meg videre.

Simen Fredrik Brunvand Fredriksen
Bachelorprosjektet har vært utfordrende og svært lærerikt. Jeg sitter igjen med en mye større forståelse og erfaring når det kommer til systemutvikling og prosjektarbeid enn det jeg hadde tidligere. Noe jeg kommer til å ta med meg videre og bruke i fremtiden. I hovedsak har jeg jobbet med frontend utvikling i React og har tilegnet meg større kunnskap innen dette feltet. Det beste med prosjektet har vært å se hvordan hele gruppen har utviklet seg til et velfungerende team.

Vebjørn Nordby
I dette prosjektet har jeg i hovedsak arbeidet mest med React og React Native da jeg allerede hadde noe kunnskap om HTML og CSS. Ettersom vi måtte tilegne oss mye ny kunnskap på kort tid var det viktig å være effektive og bygge videre på

tidligere kunnskap. Det viktigste læringsutbyttet fra prosjektet for min del har vært å se hvor viktig god kommunikasjon er i et team. Noe annet viktig jeg har lært er at det er viktig å håndtere risiko i et prosjekt. Om det finnes noen usikre elementer i et prosjekt er det lurt å ha en handlingsplan når det inntreffer.

9. Referanser

Microsoft Azure (2019) Hentet fra:
https://azure.microsoft.com/da-dk/

EVRY. (2019). Hentet fra:
https://www.evry.com/

State of JS (2018) Hentet fra:
https://2018.stateofjs.com/front-end-frameworks/overview/

Albrecht K. (2019) "*Organizational Intelligence & Knowledge Management: Thinking Outside the Silos*" Hentet fra:
http://www.karlalbrecht.com/downloads/OI-WhitePaper-Albrecht.pdf

Schwaber K. and Sutherland J. (2017) "*The definitive guide to scrum: The rules of the game.*" Hentet fra:
https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf

Robert C. Martin. (2009). "*Clean Code: A Handbook of Agile Software Craftsmanship(*utg. 1*)*". Prentice Hall.

Moløkken-Østvold, K., Haugen, N., & Benestad, H. (2008). Using planning poker for combining expert estimates in software projects. Hentet fra: *Journal Of Systems And Software*, *81*(12), 2106-2117. doi:10.1016/j.jss.2008.03.058

Clockify (2019) Dashboard. Hentet fra:
https://clockify.me/dashboard

Sam Guckenheimer (2017). *What is continuous integration?* Hentet fra
https://docs.microsoft.com/en-us/azure/devops/learn/what-is-continuous-integration

Kroenke, David M. & Boyle, Randall J. (2016) *Experiencing MIS* (6 utg.)
Pearson Education Limited

What is SMART in Project Management?. (2019). Hentet fra:
https://www.wrike.com/project-management-guide/faq/what-is-smart-in-project-management/

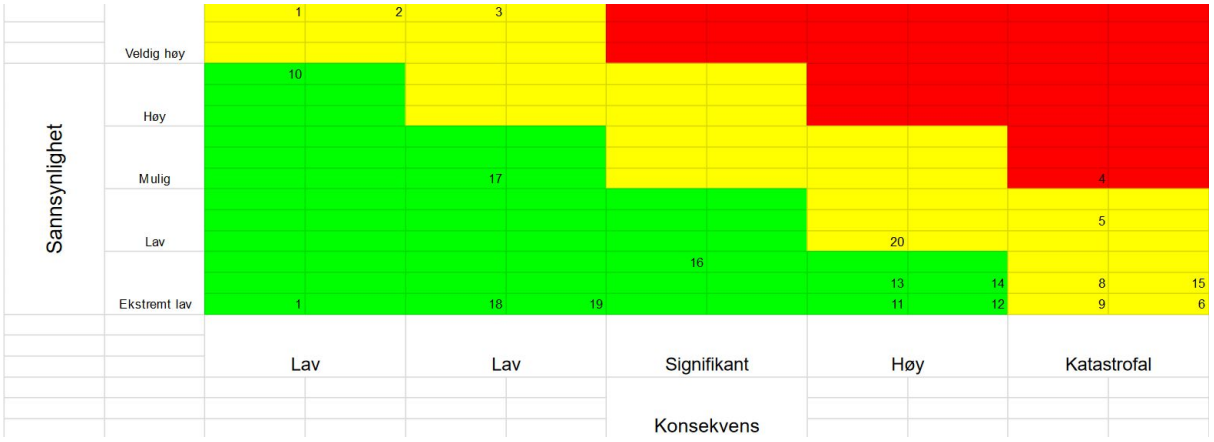
Mountain Goat Software. (2019) Scrum. Hentet fra
https://www.mountaingoatsoftware.com/agile/scrum

9.1 Figurer

10. Vedlegg

Innholdsfortegnelse vedlegg

Vedlegg 1 Risikomatrise og analyse



Risiko	Konsekvens: 1-5	Sannsynlighet: 1-5	Risikovurdering	Tiltak for forebygging	Tiltak for skade minsking
1. Ingen tilgang til å fjerne bookinger som ikke er blitt verifisert	3	5	15	God kommunikasjon og komme med løsninger som kan fikse dette problemet. Lage en fake bruker i booking som heter Nimbook der vi kan endre felt ikke verifisert og verifisert.	Ordne møte med produkteier hvor vi diskuterer situasjonen og eventuelle løsninger.
2. Lettere sykdom i gruppen	1	5	5	Vanlig god hygiene i hverdagen både på jobb og hjemme. Ikke la en person stå med all kunnskap om spesifikke oppgaver .	Om man er syk kan man jobbe hjemme og kommunisere med gruppen via sosiale verktøy.
3. Eksamen i andre fag tar tid bort fra	2	5	10	Estimere tid til eksamensøving i eksamensperioden.	Forlenge arbeidstiden i eksamensperioden eller flytte

prosjektet				Være frampå med arbeid før eksamenstid.	eksamensøving til bare i vår fritid.
4. Misforståelse mellom produkteier og gruppa	5	3	15	Ha god og kontinuerlig kommunikasjon med produkteier.	Kalle inn til et ekstra møte så fort som mulig for å oppklare misforståelser.
5. Deltidsjobber kommer i veien for arbeidstid	2	5	10	Holde gruppen oppdatert på arbeidstider i deltidsjobber.	Takke nei til ekstra jobb om det er mye å gjøre på oppgaven.
6. Autentisering mot EVRY sin AD fungerer ikke	5	1	5	Lese oss opp på hvordan denne autentiseringen har blitt brukt før. Si ifra tidlig med problemer som oppstår underveis. Bruke hjelpen vi kan få fra EVRY.	Bruke en annen autentisering, minuset kan være at de må lage seg nye brukere.
7. For mye pausebruk	1	5	5	Holde tiden på avtalte pauser.	Ta igjen tapt arbeidstid på fritiden.
8. Dødsfall i gruppa	5	1	5	Fremme god og sunn helse for andre medlemmer i gruppa. Oppfordre til trening og se ned på snus, alkohol, røyk og sukker.	Snakke med veileder om veien videre.
9. Utestengelse fra gruppa	5	1	5	Ha en tydelig gruppekontrakt som har klare regler på hvordan man skal jobbe og oppføre seg.	Vurdere arbeidsomfanget videre for oppgaven

10. Lege og tannlegebesøk	1	4	4	Oppdatere gruppemedlemmer om avtalte legebesøk kommende tid.	Ta igjen tapt arbeidstid på fritiden.
11. Data/arbeid går tapt	4	1	4	Ha gode rutiner på backup og lagring når man jobber.	Hente data tilbake fra tidligere versjoner i oppgaven. Estimere ekstra arbeidstid for å komme i mål.
12. Server downtime	4	1	4	Bruke anerkjente distributører av servere som microsoft eller google.	Holde brukere oppdatert på downtime, og ha en reserveløsning om langvarig nedetid.
13. Alvorlig sykdom	4	1	4	Oppdatere om helsesituasjon så fort som mulig.	Fordele arbeidsoppgaver fra den syke til de andre på gruppen.
14. Splid i arbeidsgrupp a	4	1	4	Bruke gruppekontrakten for å opprettholde god kultur i arbeidstid og fritid.	Løse konflikter før forhold blir dårligere. Referere til kontrakt. Utøve konsekvenser til brudd på regler.
15. EVRY går konkurs	5	1	5	Ha god dialog og kontinuerlig dialog med EVRY.	Kontakte veileder om videre arbeid. Eventuelt fortsette arbeid med en fiktiv produkteier.
16. Veileder har ikke tid til møte	3	2	6	Være fleksible på møtetid. Ha god og kontinuerlig dialog med veileder.	Foreslå nytt tidspunkt eller få veileder til å velge tidspunkt selv.
17. For mye småprat i arbeidstiden	2	3	6	Ta pauser om man vil prate, så man ikke forstyrrer andre.	Si ifra om for mye snakking. Foreslå pause for de aktuelle.

18. Ødelagt utstyr	2	1	2	Behandle utstyr på en god måte. Behandle de som vårt arbeidsverktøy. Lagre data i backup.	Spleise eller legge ut egne penger for å kjøpe nytt utstyr. Hente data fra backup.
19. Mister tilgang til arbeidslokale	2	1	2	Holde god kontakt med EVRY, og være oppdatert på status for arbeidslokale.	Ha en god plan b for arbeidslokale. Reservere rom på UIA i god tid.
20. Verifisering via mobilapp blir ikke ferdig	4	2	8	Være flere på den og ha en god plan på gjennomførelse.	Ha en reserveplan dersom APP ikke lar seg gjennomføre.

Vedlegg 2 Kvalitetsdokument for NimBook

For å måle kvalitet i prosjektet var det viktig å definere hva kvalitet er for produkteier. Her satte vi oss ned med produkteier tidlig i arbeidsperioden for å finne ut hva de definerte som kvalitet. Gjennom samtalen med produkteier ble vi fort klar over produkteiers synspunkter på kvalitet. De viktigste punktene som kom fram i samtalen handlet om kodekvalitet, sikkerhet, testing og mulighet for videreutvikling av systemet. Det at koden var lett å lese, at koden var testet og at koden hadde en standardisert struktur var det viktigste for produkteier. Videre var produkteier veldig opptatt av å bli med i ukentlige møter med oppdateringer og status.

Etter vi fikk definert kvalitet for produkteier måtte vi da finne ut hvordan vi skulle opprettholde en god kvalitetskontroll. Definisjonen av kvalitetskontroll er; *"I Enhver prosedyre utviklet for å forbedre og vedlikeholde høyest mulig kvalitetsnivå og reduksjon av feil i en prosess"* (Thomsett, 2009). Det vi har gjort for å forhindre svekket kvalitet og for å forhindre feil, var å sette opp et kvalitetsdokument. Kvalitetsdokumentet ble brukt som en bruksanvisning på hvordan vi skulle strukturere og sette opp kode til prosjektet. Ved hjelp av et kvalitetsdokument kan hvem som helst sette seg ned å forstå hvordan og hvorfor vi har strukturert koden slik vi har. En annen grunn for at vi produserte dette dokumentet var at vi som en gruppe alltid kunne følge en mal på hvordan vi skulle utvikle produktet. Dokumentet vil igjen fremme tilgjengelighet for videreutvikling av andre utviklere som ikke har

jobbet i prosjektet fra før av. Målet med dokumentet er at det skal se ut som det er samme person som har produsert all koden.
Samtidig som en bruksanvisning for kode vil dokumentet beskrive hvordan vi involverte og informerte produkteier underveis i utviklingen av produktet.

Involvering av produkteier

For å sørge for en kontinuerlig forbedring av kvalitet og utvikling bestemte vi oss for å involvere produkteier og potensielle brukere til å aktivt ta del i besluttninger og viktige avgjørelser. Produkteier ble invitert til hver sprint planning og retrospect og fikk ta del i sentrale avgjørelser og retningsvalg videre i prosessen. Samtidig har vi rapportert tidsbruk, status på brukerhistorier endringer i risikoanalyse og utfordringer fortløpende gjennom hele arbeidsperioden.

FAT-møte

Mot slutten av prosessen utførte vi en Factory Acceptance Tester sammen med produkteier. Formålet var å teste akseptansetestene til alle de brukerhistorier som var ferdige for å avdekke om de utførte sine oppgaver ut fra kriteriene vi har satt. Her fikk vi tilbakemelding på punkter vi måtte rette opp i. De testene som ikke fikk godkjennelse gjennom første FAT-møte rettet vi opp og fikk godkjent på det neste FAT-møte.

Risikovurdering

Vi utarbeidet en risikomatrise som tok for seg sannsynlighet og konsekvenser ved potensielle hendelser. Her la vi også inn tiltak for å forhindre og redusere skaden av disse. Dette viste seg å bli et nyttig hjelpemiddel til å vise produkteier hvilken potensielle farer vi sto ovenfor, og nyttig for oss for å vite hvilke tiltak vi muligens står ovenfor.

Dokumentasjonsstruktur

Før vi startet med utviklingen ble vi enige om å bruke JSDoc som en mal på dokumentasjon i koden. Hovedgrunnen for at vi valgte nettopp JSDoc var for at hele prosjektet er skrevet i javascript, men også siden vi har erfaring med JavaDoc fra før av. JSDoc og JavaDoc har fellestrekk som gjør det lettere for oss å dokumentere kode. JSDoc har et stort bibliotek for dokumentasjon av kode som er skrevet i Javascript. Her får vi en felles standard alle i gruppen kan følge når de dokumenterer egen kode.

Kodestandard

Ved å følge kodestandarden vi har satt får koden en gjennomgående stil som reduserer hvor tydelig det er at koden er produsert av flere utviklere. Dette gir kvalitet i den form at andre kan sette seg inn i koden og forstå hvordan koden fungerer. Under er et utdrag av hvordan vår standard tar for seg enkelte elementer.

Variabler
- camelCase
- Engelsk
- Logiske og forklarende navn

Metodenavn
- Engelsk
- camelCase
- Liten forbokstav
- Logiske og forklarende navn

Klasser/Komponenter
- PascalCase
- Logiske og forklarende navn
- Kun bokstaver, ingen tegn

Dokument
- Bruk konstanter som testverdier, ingen hardkodede verdier
- Linjene burde være nok til at man skal slippe å scrolle horisontalt på en 13/14 tommers skjerm.
- Minimalt med kommentarer til kode.

Retningslinjer for kodegjennomgang:

- Ikke sitt mer enn 60 minutter med én kodesnutt. En kan selvfølgelig bruke mer enn en time på et stykke kode, men siden konsentrasjonsevnen reduseres kraftig over tid er det lurt å ta hyppige pauser.
- Ikke review mer enn 200 linjer med kode om gangen.
- Gjerne ta i bruk mer uformelle former for code reviews som for eksempel “over-the-shoulder”, og sende filen(e) med kode til en annen person i gruppen som kan ta en titt.
- Bruk god tid på review. I stedet for å måtte gå over kode flere ganger fordi man var for rask første gangen er det bedre å være grundig når man ser over arbeid. Når dette er sagt så burde man ikke sitte for lenge heller.

- Bruke en sjekkliste for å vite nøyaktig hva man skal se etter i andre sin kode. Denne sjekklisten skal blant annet inkludere punktene fra kodestandard.

Sjekkliste for kodekontroll

- **Kodestandard/formatering:** kort fortalt det som står i “Kodestandard”.
    - Sjekk ‘variabler’, ‘metodenavn’, ‘klasser/komponenter’ og ‘dokument’.

- **Ikke-funksjonelle krav:**
    - Lesbarhet: koden er generelt lett å lese gjennom. Gangen i koden er oversiktlig.
    - Testbarhet: koden gjør det mulig å teste hver metode/funksjon. Enkelt å implementere enhetstesting.
    - Debugging: Under utvikling skal verdiene til variablene skrives ut slik at man ser hvordan variablene blir endret under kjøretiden.
    - Feilhåndtering: Mindre feil, f.eks. at brukeren skrev bokstaver i stedet for tall, skal ikke føre til at programmet krasjer, men heller en feilmelding som blir logget/skrevet ut til brukeren.
    - Sikkerhet: sensitiv informasjon skal ikke ligge i koden (f.eks. passord til server). Tilkoblingen mellom klient og server skal være kryptert (kan testes med Wireshark).

- **Test funksjonalitet:** (Metoder er funksjoner tilknyttet objekter). Test om funksjonene gjør det de skal uavhengig av input-verdier (dersom verdiene er feil skal dette håndteres med exception).
- **Rengjøring:** slette kode som er kommentert ut, i tillegg til overflødige kommentarer.

Mappestruktur

For å ha et ryddig og oversiktlig prosjekt som skal være lett å jobbe med har vi hatt fokus på vår mappestruktur. Her har vi skilt mellom frontend og backend kode. Dette er for å holde oversikt over pakker og kode som er brukt. Vi har en egen “node_modules” og “package.json” fil for frontend og backend som gir oversikt over pakker som er brukt til hvert av rammeverkene. Fordelen med å følge en mappestruktur er at alle som utvikler kode skal kunne navigere seg rundt til forskjellige deler av prosjektet. Her vil også personer som ikke har jobbet på prosjektet kunne sette seg inn det og forstå vår mappestruktur.

Grenmodell og versjonskontroll

I større prosjekter med flere deltakere er det viktig å ha god versjonskontroll og en bestemt grenmodell alle følger. Tidlig i planleggingsfasen ble vi enige om en bestemt måte å drive versjonskontroll på. Siden vår erfaring og kunnskap om versjonskontroll er lav bestemte vi oss for å bruke et hjelpemiddel som heter git-flow. Det er et program som hjelper oss å handtere versjonskontroll på en standardisert måte. git-flow er en idé om git-arbeidsflyt som dikterer hvilke grener som blir laget og hvordan de skal kobles sammen (GitFlow, 2018). Verktøyet styrer vi gjennom kommandolinjen til å sette opp nye grener, koble sammen grener og slette gamle grener. Ved hjelp av git-flow forsikrer vi oss om at alle følger samme oppskrift på hvordan grenene behandles, slik at vi unngår uoversiktlig grenstruktur. Vi valgte git-flow fordi det var enkelt å sette opp og ta i bruk. Vi så at vi ville spare mye tid ved å bruke et verktøy og samtidig hindre mange konflikter ved kobling av grener, som vi i tidligere prosjekter har hatt mange problemer med. Vi deler opp grenene i de 4 forskjellige kategoriene master, develop, feature og hotfix.

Under hele utviklingen av Nimbook bruker vi pull requests for å slå sammen kode mellom grenene. Dette vil da si at minst 2 andre utviklere må gå gjennom den kommende endringen for å se etter feil eller mangler før grenene blir sammenslått. Dette fører da til at vi forhindrer at kvalitet går tapt når ny kode legges til i produktet.

Master

Master grenen inneholder ferdig kode som er klar for utgivelse. Her endres ikke koden utenom når vi kommer med nye forbedrede oppdateringer på systemet. Eneste måten vi kan fikse eventuelle feil på er ved å lage en hotfix gren.

Hotfix

Hotfix grener brukes til å hurtigpatche produksjonsutgivelser. Hotfix-grener er ganske lik som develop og feature grener, men med unntak av at de er basert på master i stedet for develop.(GitFlow, 2018)

Develop

Develop grenen hentet fra master og er den grenen vi har fungerende og ferdig kode som enda ikke er klar for utgivelse. I stedet for en enkelt master gren bruker git-flow arbeidsflyten to grener til å registrere prosjektets historie. Master grenen lagrer den offisielle utgivelsesloggen, og develop grenen fungerer som en integrasjons gren for funksjoner fra feature grenene (GitFlow, 2018).

Feature

Feature grenene er der vi jobber og utvikler ny kode. Feature henter direkte fra develop grenen. Denne innkapslingen gjør det mulig for flere utviklere å jobbe med en bestemt funksjon uten å forstyrre hovedkoden. Det gjør at master aldri vil inneholde ødelagte koden, noe som er en stor fordel for kontinuerlige integrasjonsmiljøer (GitFlow, 2018).

Test plan

Oppdragsgiver har lagt stor vekt på testing av kode. Hensikten med testing av kode er å forsikre at koden oppfører seg som forventet ved endringer på senere tidspunkt. Når man har fullført en kodesnutt skriver man en test på denne, noe som forsikrer om at koden gjør det som er forventet. Det viktigste med testing er dog når man går tilbake for å optimalisere hastighet eller lesbarhet for koden, da forsikrer testene seg om at koden oppfører seg som forventet og at det ikke blir introdusert noen nye feil

ved faktureringen. Testing av kode gir ingen garanti i seg selv om høy kvalitet, men det hjelper til å styrke den totale kvalitetsfølelsen ved å introdusere en høy grad av tiltro. Hvis en av testene feiler blir førsteprioritet å få fikset feilen.

Sikkerhet

Personvernforordningen som trådte i kraft 25 Mai 2018 er en forordning som satte strenge krav og regler på hvordan medlemslandene i EU skal behandle persondata og personvern. Det er derfor viktig for EVRY at vi holder oss oppdatert på de nye forordningene om dette skal være et system vi skal levere til de. I Nimbook skal vi ikke lagre noen form for data, men vi henter data fra eksterne APIer. Det er da viktig at vi har en form for sikkerhet og autentisering i produktet. Vi løser dette med at brukere logger seg inn gjennom Microsoft med sine egne brukere og godkjenner tilgang. Da slipper vi å lagre data om brukere. Brukere kan først bruke Nimbook etter de har autentisert seg.

Vedlikeholdbarhet

Vedlikeholdbarhet beskrives som høy kvalitet for EVRY. Det at hvem som helst skal kunne gå inn i vårt prosjekt i ettertid for å videreutvikle eller drive vedlikehold uten tidligere erfaring med prosjektet er noe som styrker kvaliteten i stor grad. Dette oppnår vi ved å vise til strukturer og standarder vi har benyttet oss av gjennom prosjektet, så han eller hun som skal gå inn i vårt prosjekt skal kunne se hvordan vi har valgt å utforme vår mappe/fil struktur og kode. Selve mappestrukturen har vi fordelt i frontend og backend mapper. Hvor alle pakker som er brukt ligger i sin tilhørende mappe slik at det er lett å skille mellom tilhørighet.

For å holde koden ryddig og lesbar har vi brukt Prettier til å formatere koden basert på regler som vi har satt i vår kodestandard. Dette gjør koden lettere å lese da selve strukturen og utseende ser likt ut i alle filer. I tillegg skal all kode være kommentert slik at man kan lese kommentarene og forstå gangen i koden.

For å opprettholde høy kvalitet og vedlikeholdbarhet har vi et sett regler for godkjenning av pull requests. Her må to personer i gruppen lese gjennom koden og se at den er lett leselig, ikke for mye kode på hver side, og at all kodestandard er opprettholdt. Dette bidrar til høy kvalitet, og gjør at systemet er lettere å vedlikeholde (Yu, Y., Wang, H., Yin, G., & Wang, T., 2016, s. 1).

Vedlegg 3 **Intervju med Christian Moen**

Besvarelse - Christian (Evry #1)
Generelt

1. **Har dere en løsning for møteromsbooking?**
   Hvis ja, hvilken løsning bruker dere?
   Outlook møte hvor man legger til rommet som deltaker i rommet.

2. **Er det noe som oppleves som positivt med booking-løsningen dere bruker i dag?**
   Kommer rett i kalender, kan invitere fra Outlook
   Oversikt over ledighet i kalender

3. **Er det noe som oppleves som negativt med booking-løsningen dere bruker i dag?**
   Overbooking
   Vanskelig å se om det er ledig når man booker rommet, må åpne kalenderen og spesifikt sjekke det aktuelle rommet.
   Ingen god oversikt over når det er ledig, må sjekke alle rom

4. **Er det noe som oppleves som "tungvint" når en skal booke grupperom?**
   Skrive inn når man ønsker møte, få igjen forlag om når rom er tilgjengelige
   Hvor mange mennesker man skal ha med i møte, best egnet.

5. **Er det noe du føler mangler i bookingsystemet som brukes i dag?**
   Intuitivt
   Så få klikk som overhodet mulig, må være veldig enkelt

6. **Hvordan er det perfekte for deg? Hvilke funksjoner inngår i dette systemet?**
   Finner automatisk det beste rommet ut ifra satte kriterier.
   Finner det nærmeste rommet som passer kriteriene.

7. **Tenker du det er enklest å booke møterom fra app på mobil, eller nettleser?**
   Mulighet for bruk av både nettleser og mobil for å booke rom, ønsker ikke å bli bundet til en satt platform.

8. **Hva gjør du om det ikke er noen ledige rom på skolen/arbeidsplassen?**
   Sjekker etasjen under, tar opp unødvendig tid.
   Må gå fysisk rundt for å se om det er ledige rom, tar tid og er ikke veldig effektivt mtp at det er vanskelig å se inn i noen av rommene pga frostet glass og lignende.

9. **Synes du bookinger burde verifiseres?**
   Hvis ja, hvor lang tid burde det ta før reservasjonen blir kansellert?

Fint med verifisering, men må være lett å få til
Ikke alle gidder å slette booking

Verifisering/kansellering

10. **Dersom bookinger burde verifiseres; hva tror du/dere er den enkleste formen for verifisering av oppmøte på et møterom?**
    Verifisering enten via e-post eller mobil, intuitivt
    Trykke på skjerm utenfor rommet for å verifisere

12. **Hvordan kansellerer man en booking slik det er i dag?**
    Slette booking i Outlook
    lett å glemme bort

Sammendrag

Evry har per dags dato et eget system for booking av møterom. Systemet de bruker er Outlook. Det positive med å bruke Outlook er at alle bookinger havner i kalenderen, der alle andre kan få en oversikt over møterom som er opptatt. Noe annet som er positivt er at man kan legge til deltakere i bookingen. Deltakerne får da en mail om bookingen og får et valg om å bekrefte bookingen. Bekrefter de bookingen legges det automatisk inn i kalenderen på Outlook.
De negative sidene med systemet er overbooking og dårlig oversikt. Det er ingenting som stopper noen for å overbooke, og i tillegg er oversikten over møterommene dårlig. Det finnes ingen oversikt som viser alle rommenes status, så man må manuelt sjekke hvert spesifikke rom for å se booking status.
Det kan også oppleves som tungvint at det ikke finnes noen informasjon om rommene de booker. Om man er flere deltakere, finnes det ingen info om det er plass på det spesifikke møterommet. Det hadde vært fint med forslag om tilgjengelighet så man får det rommet som passer til forskjellige tidspunkt og til forskjellig mengde deltakere.
Når det var snakk om mangler på systemet var det ønsker om å gjøre systemet mer intuitivt og at det skulle kreves minde klikk for å komme i mål.
Det perfekte systemet ville ha funnet det beste og mest tilgjengelige rommet på kortest mulig tid. Der det kreves minst mulig arbeid for å få det beste rommet.
Når det var snakk om hvilken plattform de brukte på dagens system, så er det pc som blir mest brukt. Man kan bruke mobilen, men siden man alltid sitter på en datamaskin vil det være det mest praktiske. Booking med mobil går like fint, men en app hadde vært praktisk. Om det fantes skjermer på utsiden av grupperommene kunne det og være en god løsning.
På spørsmål om hva de gjør om alle rommene er booket svarer Christian at man da må fysisk gå rundt for å dobbeltsjekke om at det ikke er noen møterom som står

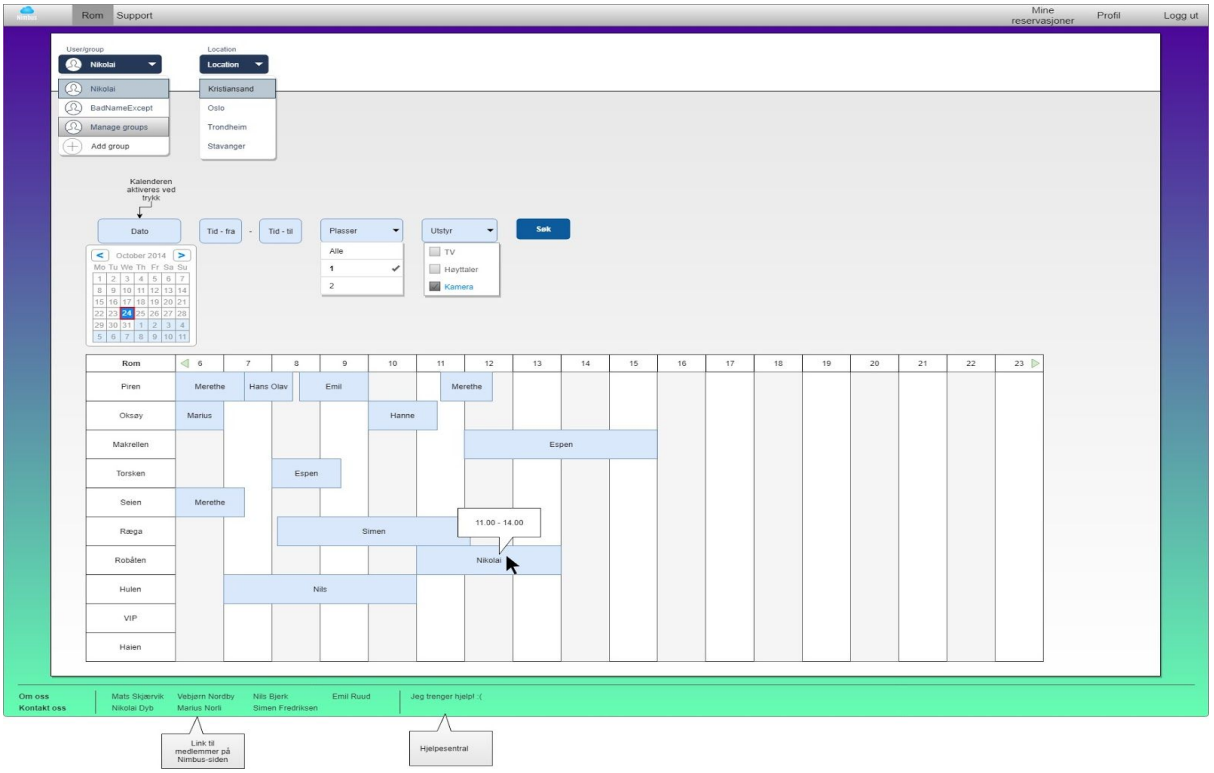
tomme. De kan eventuelt booke rom i 1 etasjen via resepsjonen. De har en “regel” på at de lar dørene på møterom stå åpen når de er ledige.
For å kansellere en booking må man manuelt gå tilbake i Outlook og gjøre det manuelt. Det er det ikke mange som gjør siden det fort kan glemmes, og at det er litt tiltak. Om det fantes en verifisering for bruk av grupperom hadde det vært velkommen. En form for verifisering via epost eller ved å trykke på en skjerm utenfor rommet kan være en praktisk løsning.

Vedlegg 4 - Brukerhistorier

Brukerhistorie 14	Som en bruker av systemet, ønsker jeg å bli varslet og få informasjon om møter jeg blir invitert til, slik at jeg kan forberede meg til møtet.
MoSCoW	**S**
Beskrivelse	Når en bruker blir invitert/lagt til i en møteromsbooking, vil man få et varsel og informasjon om møtet. Dette gir brukeren muligheten til å vite hva møtet handler om.
Begrunnelse og rangering	Tilbakemeldingene vi har fått i intervjurunden har vært at bruker ønsker en varsling med informasjon om møtene. Informasjonen skal være såpass utfyllende at bruker kan forberede seg til møtet.
Akseptansekriterier	Når en bruker blir invitert til et møte, skal denne brukeren få et varsel fra systemet om dato, tid, hvem som deltar på møtet, og hva møtet går ut på.
Akseptansetester	**Akseptansetest 1** 1. Bruker blir invitert til et møte. 2. Bruker mottar en varsel/mail. 3. Bruker ser informasjon om møtet og tidspunkt.

Brukerhistorie 28	Som en bruker, ønsker jeg mulighet for å booke et rom i en satt tidsperiode, slik at rommet er mitt i den perioden jeg booker det.
MoSCoW	**M**
Beskrivelse	Brukeren kan booke rommet i en tidsperiode hvor det er ledig
Begrunnelse og rangering	Det å kunne booke et rom er en vesentlig funksjon i vårt system. Når systemet skal håndtere bookinger er det viktig at den kan lage nye bookinger.
Akseptansekriterier	Når en bruker blir invitert til et møte, skal denne brukeren få et varsel fra systemet om dato, tid, hvem som deltar på møtet, og hva møtet går ut på.
Akseptansetester	**Akseptansetest 1** 1. Bruker ser oversikt over rom 2. Bruker velger tidsperiode for bookingen 3. Bruker booker rommet

Vedlegg 5 Wireframe



Vedlegg 6 FAT - Factory Acceptance Test

Godkjent	Mangelfull
[ ]	[x]

Bruker historie	Akseptansetest	Status	Kommentar
1	1. Bruker logger inn på systemet 2. Systemet sjekker om bruker er lagret 3. Bruker blir logget inn i systemet 4. Systemet henter informasjon og viser	OK	Vi har nådd målet Storyen: akseptansetesten og storyen henger ikke sammen. Vi er innom

	informasjon fra databasen		kalender og rom i akseptansekriteriene, når det er snakk om login.
2	1. Bruker går til “domenenavn” hvor systemet er implementert. 2. Bruker klikker seg inn på oversikten over rom. 3. Oversikten viser når rommene er reservert, og når de er tilgjengelige for å bookes.	OK	Husk valg av romlister
3	1. Bruker går til “domenenavn” hvor systemet er implementert. 2. Bruker klikker seg inn på oversikten over rom. 3. Bruker klikker på møterom som allerede er reservert på et bestemt tidspunkt. 4. Bruker prøve å booke rommet. 5. Systemet foretar en sjekk på om den forespurte bookingen finner sted samtidig med en annen booking. 6. Bruker får beskjed om at det ikke er umulig å reservere møterommet i det etterspurte tidsrommet.	Ikke ok	Vi får ikke booket via oversikter. Bug som viser at første rom kommer uanset hvilket rom du prøver å booke.
4	1. Bruker går inn på siden. 2. Bruker fyller inn passord og mail 3. Bruker trykker på logg inn. 4. Bruker får tilgang til systemet hvis påloggingsinformasjon er korrekt.	Ok	Samme som brukerhistorie 1
7	1. Bruker går til “domenenavn” hvor systemet er implementert. 2. Bruker klikker seg inn på oversikt over rom. 3. Bruker klikker på ønsket møterom. 4. Bruker booker møterom på ønsket tidspunkt. 5. Brukeren får bekreftelse via mail.	Ikke ok	Bug på Rom liste som gjør at bare det øverste rommet viser

8	1. Bruker går til “domenenavn” hvor systemet er implementert. 2. Bruker klikker seg inn på oversikten “Mine reservasjoner”. 3. Bruker klikker "avlys" på ønsket reservasjon. 4. Brukeren får opp en advarsel på kansellering. 5. Brukeren bekrefter ved å klikke på “Avlys”.	Ikke ok	Får ikke frem mine reservasjoner. Fungerer ikke.
28	1. Bruker ser oversikt over rom 2. Bruker velger tidsperiode for bookingen 3. Bruker booker rommet	Ok	Ingenting å påpeke
10	1. Bruker klikker seg inn på oversikt over rom. 2. Bruker ser kapasitet for hvert enkelt rom på romoversikt.	Ok	Ingenting å påpeke
16	1. Bruker blir invitert til et møte. 2. Bruker mottar en varsel/mail. 3. Bruker ser informasjon om møtet og tidspunkt.	ok/hmm	Vi må få til location smutthull. Godkjent, men ønsket forbedring.

Antall godkjente	6 / 9

Vedlegg 7 Gruppekontrakt

Nimbus

Gruppekontrakt
Vår 2018
IS-304
Bachelorprosjekt - Universitetet i Agder, Evry

Denne kontrakten er gjeldene for medlemmene av gruppen Nimbus under gjennomføringen av en bacheloroppgave ved Universitetet i Agder, i samarbeid med Evry Kristiansand.

Medlemmer

Navn	**E-post**
Marius G. Norli	marign16@uia.no
Emil M. Ruud	emilr15@uia.no
Nils Fredrik I. Bjerk	nfbjer16@uia.no
Vebjørn Nordby	vebjon16@uia.no
Mats E. Skjærvik	matses15@uia.no
Simen Fredrik B. Fredriksen	sffred16@uia.no
Nikolai Holmen Dyb	nikold16@uia.no

Krav
Alle medlemmer plikter til å utføre sin del av arbeidet, samt å møte opp til avtalte tider med mindre noe uforutsett skulle skje.

Mål
Målet med prosjektet er først og fremst å utvikle et system arbeidsgiver er fornøyd med. Samtidig ønsker vi som en gruppe å bruke denne tiden til å tilegne oss kunnskap om nye og relevante teknologier. Dette, sammen med at vi jobber oss gjennom hele livssyklusen til et systemutviklingsprosjekt, håper vi vil bidra til kompetanse som gjør oss ettertraktet på arbeidsmarkedet. Gruppen som en helhet sikter seg også inn mot høyeste mulig karakter.

Arbeidsmengde
Mengden arbeid har vi beregnet ut ifra hva en total arbeidsuke inneholder, delt på antall studiepoeng for faget. Vi har derfor kommet

frem til at 10 studiepoeng er tilsvarende 13 timer. Med tanke på at et semester er 30 studiepoeng har vi beregnet 37,5 timer for prosjektet i sin helhet hvorav 26 timer er allokert til IS-304 og 11,5 timer er allokert til IS-305.

Arbeidsmetode

Gruppen ønsker å arbeide i Evry sine lokaler på Kjøita fra Mandag-Torsdag, 9:00-15:30 med faget IS-304. Fredager og noe kveldsarbeid er beregnet på faget IS-305. Vi har også bestemt oss for at forelesninger, samt transport mellom Evry og forelesninger er godkjent som arbeidstimer.

Konsekvenser

Gruppemedlemmene kjenner hverandre godt fra før av, men har samtidig blitt enige om at problemer med oppmøte burde medføre en straff. Vi har derfor kommet frem til følgende regler:

- Dersom et medlem av gruppen ikke er tilstede på kontoret før kl 09:00 selv ved gitt beskjed, er han skyldig 2 stk *butikkpils* felles til gruppen.
- En butikkpils er definert som 1 stk 0,5l eller 0,33l flaske pilsener av valgfritt slag.
- Dersom man i tillegg ikke sier ifra om forsentkommelse, legges 1 stk *butikkpils* til hvor hver time gruppemedlemmet ikke møter opp. Eks ved 2 timer forsentkommelse vil medlemmet være skyldig 4 stk *butikkpils*.
- Dersom et gruppemedlem ikke møter opp ila en hel arbeidsdag uten å gi beskjed i god tid, blir medlemmet som kommer for sent skyldig i 1 stk *utepils* per gruppemedlem.
- En utepils er definert som en enhet kjøpt på et utested.
- Alle disse reglene *kan* bli unntatt dersom noe ekstraordinært og uforutsett skulle inntreffe, medlemmet plikter da å si ifra i så god tid som mulig.

Signaturer

……..
Marius Granum Norli

……..
Emil Morønning Ruud

……..
Vebjørn Nordby

……..
Nils Fredrik Iselvmo Bjerk

……..
Mats Eide Skjærvik

……..
Nikolai Holmen Dyb

……..
Simen Fredrik Brunvand Fredriksen

Litteraturliste

GitFlow (2018) Hentet fra: https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow

Michael C. Thomsett. (2009) *The little black book of project management* (3 utg.). American Management Association.

Yu, Y., Wang, H., Yin, G., & Wang, T. (2016). Reviewer recommendation for pull-requests in GitHub: What can we learn from code review and bug assignment?. *Information And Software Technology*, *74*, 204-218. doi:10.1016/j.infsof.2016.01.004