

**IDATT2106
Sluttrapport
For Krisefikser
Versjon 1.0**

Teammedlemmer

Scott du Plessis, Aryan Malekian, Jonathan Hubertz, Sander Nessa, Usman
Ghafoorzai, Sander Berge, Mikael Stray

Produkteiere

Kristian Storehaug, Zakariya Ibrahim

Revisjonshistorie

Dato	Versjon	Beskrivelse av endring	Forfatter
14.05.2025	1.0	Første utkast	Team 10
15.05.2025	1.1	Ferdig versjon	Team 10

Sammendrag	3
Forord	4
Oppgavebeskrivelse	4
1. Introduksjon	5
1.1 Hensikt og bakgrunn	5
1.2 Innhold og struktur	5
1.3 Akronymer og forkortelser	5
2. Teori og relevant litteratur	6
2.1 Forhåndskunnskap og erfaringer	6
2.2 Smidig utvikling og Scrum	7
2.3 Sprints	7
2.4 Scrum roller	7
2.4.1 PO	7
2.4.2 Team	7
2.4.3 Scrum Master	8
2.5 Product Backlog og prioritering	8
2.6 Sprint Planning – planlegging av arbeidet	8
2.7 Sprint Backlog – Konkret arbeidsliste	8
2.8 Daily Scrum – daglig statusmøte	9
2.9 Burndown Chart – visuell fremdrift	9
2.10 Sprint Review – tilbakemelding på leveranser	9
2.11 Sprint Retrospective – kontinuerlig forbedring	9
2.12 Brukertest	10
2.13 Universell utforming	10
2.14 Personvern og GDPR	10
2.15 Sikkerhet	11
2.16 Testing	11
2.16.1 Enhetstesting	11
2.16.2 Integrasjonstesting	11
3. Metode	12
3.1 Scrum	12
3.2 Seremonier	12
3.2.1 Daily scrum	12
3.2.2 Sprint Planning	13
3.2.3 Sprint Review	13

Scrumteamnr 10	Versjon: <1.0>
Sluttrapport	Dato: <15/05/25>
dokumentidentifikator	

3.2.4	Sprint retrospekt	13
3.3	Artefakter	14
3.4	Systemarkitektur og design	14
3.4.1	Overordnet arkitektur	14
3.4.2	Teknologier brukt	15
3.5	Testing	15
3.5.1	Brukertesting	15
3.5.2	Automatisk testing av programvare	16
3.6	Personvern, GDPR og sikkerhet	16
3.6.1	GDPR og personvern	16
3.6.2	Sikkerhet	16
3.7	Universell utforming	17
3.7.1	WCAG og WAVE	17
3.7.2	CI/CD	17
4.	Resultater	18
4.1	Prosjektresultater	18
4.1.1	Måloppnåelse Sprint 1	18
4.1.2	Måloppnåelse Sprint 2	18
4.1.3	Tilbakemeldinger på brukertester	18
4.2	Overordnet måloppnåelse i forhold til visjonsdokumentet	19
4.3	Administrative resultater	20
4.3.1	Sprint Retrospektiv	20
4.3.2	Samarbeid og teamutvikling	20
4.3.3	Roller og ansvar	20
4.3.4	Artefakter	21
5.	Konklusjon og videre arbeid	21
5.1	Oppsummering av prosjektets sluttresultat	21
5.2	Evalueringsprosess og metode	21
5.3	Refleksjon over teknologisk valg	22
5.4	Læringsutbytte og teamutvikling	22
5.5	Forbedringspunkter og anbefalinger	22
	Bibliografi	23

Sammendrag

Denne rapporten viser utviklingen av Krisefikser – en applikasjon med mål om å styrke egenberedskapen i Norge. I en digital verden er det nødvendig med et digitalt verktøy for å forberede seg før, under og etter en krise. I et håp om å gjøre dette mer tilgjengelig er det blitt utviklet en applikasjon som kombinerer alt man trenger for krisehåndtering. Krisefikser bygger videre på tidligere tiltak innenfor fagfeltet som informasjonsspredning, ved å tilby en brukervennlig og engasjerende applikasjon for læring og beredskap. Om du er usikker på hvordan du forbereder eller håndterer en krise, da er Krisefikser noe for deg.

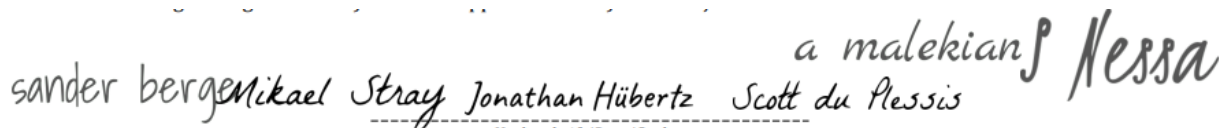
Scrumteamnr 10	Versjon: <1.0>
Sluttrapport	Dato: <15/05/25>
dokumentidentifikator	

Utviklingsarbeidet ble gjennomført gjennom to sprinter, som ga et funksjonelt produkt. Krisefikser resulterte i en sikker og funksjonell applikasjon med funksjoner for innlogging, beredskapslager, kartvisning og interaktiv quiz. Krisefikser har tydelig fokus på universell utforming og brukerens personvern. Prosjektet bygger videre på eksisterende tiltak innen beredskap, som er preget av statlig informasjonsdeling og veiledning. Krisefikser har et mål om å videremidle relevant informasjon i en interaktiv og brukervennlig applikasjon. Gjennom smidig utvikling, er det utviklet en komplett applikasjon som skal gjør deg engasjert for egenberedskap. Denne rapporten viser prosessen, metoden, resultatene og erfaringene fra utviklingen av Krisefikser – en enkel løsning for egenberedskap i din hverdag.

Forord

Denne rapporten markerer slutten på utviklingen og samarbeidet av Krisefikser – en applikasjon for å styrke egenberedskapen for folk flest. Oppgaven ble valgt på bakgrunn av en tydelig utfordring: Hva gjør man om krisen inntreffer? Det er et stort behov for økt kunnskap og bedre digitale verktøy for krisehåndtering. Gjennom en smidig utviklingsprosess og samarbeid med produkteiere har det blitt utviklet en løsning som skal svare på problemene.

Arbeidet har vært lærerikt og utfordrende, og teamet tar med verdifull erfaring fra teamarbeid med smidig systemutvikling. En spesiell takk rettes til Pedro Pablo Cardona Arroyave for teknisk støtte og god veiledning gjennom hele utviklingsløpet.



Underskrift/Dato/Sted
 15. Mai 2025

Oppgavebeskrivelse

Prosjektoppgaven gikk ut på å utvikle en webapplikasjon som skal bidra til å øke egenberedskapen blant Norges innbyggere. Dette innebar funksjonalitet for lageroversikt, generell kriseinformasjon og kartbaserte tjenester. Oppgaven og relevante effektmål er beskrevet i visjonsdokumentet, som ligger i vedlegget. Underveis i utviklingsprosessen var det nødvendig, i samråd med produkteiere, å prioritere de viktigste effektmålene. Dette medførte at enkelte funksjoner ble nedprioritert, mens andre ble prioritert. Underveis har teamet prioritert hovedfunksjonene og interaktive elementer som quiz. Dette har bidratt til en løsning som teamet mener er målrettet, funksjonell og brukervennlig.

Scrumteamnr 10	Versjon: <1.0>
Sluttrapport	Dato: <15/05/25>
dokumentidentifikator	

1. Introduksjon

Dette dokumentet er sluttrapporten for Krisefikser prosjektet, gjennomført som en del av emnet IDATT-2106-Systemutvikling med smidig prosjekt, ved NTNU våren 2025. Rapporten dokumenterer arbeidet og prosessen til Scrumteam 10, som i samarbeid med produkteiere fra masterprogrammet "Digital Transformasjon" har utviklet en webapplikasjon for håndtering av egenberedskap for før, under og etter en krise.

1.1 Hensikt og bakgrunn

Bakgrunnen for prosjektet er et reelt samfunnsbehov: Hva gjør man om en krise inntreffer? Nordmenn flest er ikke godt nok forberedt på mulige krisesituasjoner som naturkatastrofer, langvarige strømutbrudd eller forsyningssvikt. I lang tid har direktoratet for samfunnssikkerhet og beredskap (DSB) anbefalt at alle husstander skal være utstyrt med et beredskapslager som kan dekke behov om krisen inntreffer. For de fleste kan slik informasjon være overveldende, og det er derfor et behov for en applikasjon som gjør det enklere for folk flest å gjøre forberedelser og øke sin kunnskap om generell krisehåndtering.

Oppdragsgiveren har et ønske om at det skal utvikles et produkt med effektmål om å styrke bevisstheten og kunnskapen rundt beredskap. Krisefikser skal ikke bare informere, men også motivere og engasjere brukeren til å lære om krise og beredskapshåndtering på en brukervennlig måte.

1.2 Innhold og struktur

Gjennom rapporten skal leseren få en innsikt i arbeidsprosessen til teamet og hvordan det har blitt utviklet en brukervennlig og samfunnsnyttig applikasjon som gir en løsning på reelle behov. Gjennom hele prosjektet har det blitt kombinert teknisk kunnskap med prinsippene for smidig utvikling. Resultatet er en applikasjon som kan gjøre egenberedskap enklere og mer tilgjengelig for den norske befolkningen.

1.3 Akronymer og forkortelser

API – Application Programming Interface

PO – Product Owner

REST – Representational State Transfer

OOP – Objektorientert programmering

URI – Uniform Resource Identifier

SM – ScrumMaster

GDPR – General Data Protection Regulation

Scrumteamnr 10	Versjon: <1.0>
Sluttrapport	Dato: <15/05/25>
dokumentidentifikator	

WCAG – Web Content Accessibility Guidelines

XSS – Cross Site Scripting

OWASP – The Open Web Application Security Project

ORM – Object Relational Mapping

NVD – National Vulnerability Database

MVP – Minimum Viable Product

JPA – Java Persistence API

HTTP – HyperText Transfer Protocol

2. Teori og relevant litteratur

2.1 Forhåndskunnskap og erfaringer

Teamet har tidligere erfaringer med OOP fra emnene *IDATT1003 Programmering 1* og *IDATT2003 Programmering 2*. Fra emnene har teamet lært om konsepter som aggregering, kohesjon, kobling, innkapsling, arv og polymorfisme. (Vihovde, 2025)

Aggregering beskriver en struktur der et objekt kan inneholde eller eie et annet, og brukes for å modellere relasjoner i komplekse systemer. Kohesjon viser til hvor godt elementene i en klasse henger sammen; høy kohesjon innebærer at klassen har ett klart ansvar. Lav kohesjon fører til uklare roller og økt kompleksitet. Kobling handler om graden av avhengighet mellom klasser; løs kobling gir mer fleksibilitet og enklere vedlikehold, mens tett kobling øker sårbarhet for endringer. Innkapsling skjuler interne data og eksponerer kun kontrollerte grensesnitt, noe som beskytter objektenes tilstand. Arv og polymorfisme gjør det mulig å gjenbruke og utvide funksjonalitet. (Geeks for geeks, 2023).

Når det gjelder å sette opp database, har teamet erfaring fra emnet *IDATT2002 Databaser*. Dette emnet satte et søkelys på datamodellering, relasjonsdatabaser, normalisering, transaksjonshåndtering og integrasjon mellom databaser og applikasjonslag.

Fra emnet *IDATT2015 Full-stack applikasjonsutvikling* har teamet erfaring med REST. Det er en arkitekturstil som bygger på prinsippene om ressurser, representasjoner og standardiserte HTTP-metoder (GET, POST, PUT, DELETE). REST legger til rette for løst koblede systemer, hvor hver ressurs identifiseres av en URI og manipuleres via veldefinerte operasjoner. Design av REST-fulle API-er krever forståelse for både programstruktur og datastruktur, og drar nytte av prinsipper som lav kobling, høy kohesjon og riktig innkapsling (Varanski & Bartkov, 2021).

Scrumteamnr 10	Versjon: <1.0>
Sluttrapport	Dato: <15/05/25>
dokumentidentifikator	

2.2 Smidig utvikling og Scrum

Scrum, som bygger på smidig utvikling, vokste frem fra eldre iterative og inkrementelle metoder (Deemer et al., 2009, s. 4). I følge Deemer et al. (2009) er kjernen i Scrum å prioritere å bygge fungerende programvare som brukere raskt kan få tilgang til, fremfor omfattende spesifikasjoner på forhånd. Det legges tyngde på tverrfaglige, selvorganiserende team, raske iterasjoner og kontinuerlig kundetilbakemelding.

Et sentralt prinsipp i Scrum er "inspisere og tilpass" (Deemer et al., 2009, s. 5). Det vil si å ta korte utviklingssteg, kontinuerlig evaluere både produkt og prosess – og deretter gjøre de justeringer som er nødvendige. Ettersom utvikling medfører læring, innovasjon og utfordringer, legger Scrum opp til en rytme av kontinuerlig forbedring.

2.3 Sprints

Iterasjonene som Scrum er basert på kalles "Sprints". Disse varer mellom 1 og 4 uker. Hver respektiv Sprint har en fast lengde og avsluttes på en bestemt dato. Ved starten av en Sprint velger teamet oppgaver fra en prioritert liste, kjent som en produktkø, og forplikter seg til å fullføre disse innen utløpsdatoen (Deemer et al., 2009, s. 4-5).

2.4 Scrum roller

Scrum definerer tre sentrale roller: *Product Owner (PO)*, *Team* og *Scrum Master (SM)* og det er verdt å merke seg – *ingen tradisjonell prosjektlederrolle*. Scrum-teamet er tverrfaglig og selvorganiserende (Deemer, Benefield, Larman Craig, & Vodde, 2009, p. 10), og har det samlede ansvaret for å levere produktet med høy kvalitet. Det består av PO, SM og utviklere, med anbefalt størrelse på 3 til 8 personer (Kniberg, 2015, p. 33).

2.4.1 PO

En PO har ansvaret for å maksimere produktets verdi. Dette gjør PO gjennom prioritering av Product Backlog. Kort skildret, er en Product Backlog en dynamisk, prioritert liste over det ønskede arbeidet for produktet. Product Backlog betyr produktkøen på norsk. Denne eies og prioriteres av PO (Deemer et al., 2009, s. 7). PO har ansvaret for å definere og kommunisere produktvisjon og foredle denne til konkrete leveranser. PO skal aktivt delta i Sprint Planning og Sprint Review for å avklare mål og gi tilbakemeldinger (Deemer et al., 2009, s. 14).

2.4.2 Team

Et Scrum utviklingsteam bør være tverrfaglig og selvorganiserende, deriblant med ansvar for å levere et potensielt "realiserbart" produkt i hver Sprint. En annen oppgave teamet har, er å estimere oppgaver, bestemme arbeidsmengde og hvordan det faktiske arbeidet skal utføres. Teammedlemmene deler ansvaret for kvaliteten – uavhengig av spesialisering

Scrumteamnr 10	Versjon: <1.0>
Sluttrapport	Dato: <15/05/25>
dokumentidentifikator	

(Deemer et al., 2009, s. 6-8; Kniberg, 2015, s. 116).

2.4.3 Scrum Master

En SM har som hovedoppgave å sørge for å bidra til at Scrum-prosessen følges – men og å fjerne hindringer og å støtte organisasjonen i å adoptere Scrum. Målet til SM er å gjøre seg overflødig ved å styrke teamets selvstendighet. Med andre ord er SM en fasilitator for teamet og PO (Deemer et al., 2009, s. 6-7).

2.5 Product Backlog og prioritering

Produktkøen er en prioritert liste bestående av alle funksjonaliteter sluttproduktet skal ha. Denne lista regnes som den eneste kilde til krav. Det er PO som ansvarlig for å prioritere disse elementene i produktkøen, og prioriteringen er ofte basert på viktighet eller verdi (Kniberg, 2015, p. 6). Teamet, på sin side, anslår arbeidsmengden (Kniberg, 2015, p. 6). Det anbefales at mellom fem - og ti prosent av hver Sprint dedikeres til å gjøre forbedringer i produktkøen, som blant annet omfatter detaljering, splitting og estimering av elementer.

2.6 Sprint Planning – planlegging av arbeidet

Sprint planning har som hovedformål å gi utviklerteamet nok informasjon om hva de skal gjøre for å kunne jobbe uforstyrret i påfølgende sprint. Deltakere i sprint planning er scrum teamet og POs, og fokuset ligger i at PO lager en prioritert liste med funksjonaliteter de ønsker at teamet skal implementere i løpet av sprinten. Deretter vil utviklerne dele opp funksjonalitetene i mindre oppgaver som de legger i sprintkøen. Under sprint planning skal man også sette sprint mål, som er et høyt-nivå mål for hva man skal få til under kommende sprint. (Kniberg, 2015)

Det er avgjørende at POs deltar i sprint planning ettersom det er de som skal sette omfang, viktighet og prioriteringer for funksjonalitet. Hvis produkteier bestemmer for å ikke delta i sprint planlegging kan man prøve å gjøre en av følgende alternativer: få POs til å forstå viktigheten av deres deltakelse i møtet, prøve å få noen andre til å delta i møtet som «PO proxy», eller be om å få tildelt nye POs eller utsette sprint planlegging til PO kan delta (Kniberg, 2015, s. 17).

2.7 Sprint Backlog – Konkret arbeidsliste

Sprint Backloggen inneholder de oppgavene teamet har forpliktet seg til å gjennomføre i sprinten, og fungerer som en operativ arbeidsplan. Den opprettes under andre del av Sprint Planning. Teamet har ansvar for å holde den oppdatert. Oppgavene kan ha estimer, som oppdateres jevnlig.

Scrumteamnr 10	Versjon: <1.0>
Sluttrapport	Dato: <15/05/25>
dokumentidentifikator	

En fysisk oppgavetavle er en visuell fremstilling av fremdriften i sprinten (sprint backloggen), ofte plassert på en vegg. Oppgaver skrives på post-it lapper som flyttes mellom kolonnene, som deles opp i for eksempel "To Do", "In Progress" og "Done" (Deemer, Benefield, Larman Craig, & Vodde, 2009, p. 10). En fysisk fremstilling fremmer samarbeid, kommunikasjon og felles forståelse innad i teamet. Selv om det kan være et digitalt dokument, anses en visuell, fysisk tavle som den mest effektive formen for Sprint (Kniberg, 2015, p. 56).

2.8 Daily Scrum – daglig statusmøte

Daily Scrum er et kort møte som avholdes hver dag på et bestemt tidspunkt, for eksempel på starten av arbeidsdagen. Fortrinnsvis skal hvert medlem dele følgende med resten av teamet: Hva de har gjort siden forrige møte, hva de har tenkt å bli ferdigstille til neste møtet, og til slutt, om det er noen problemer de sliter med. Eventuelle diskusjoner skal tas etter seremonien (Deemer et. al., 2009, s. 11, 12)

2.9 Burndown Chart – visuell fremdrift

Et burndown-diagram brukes til å følge med på hvor mye arbeid som gjenstår i sprinten, og gir en visuell indikasjon på fremdrift. Når teamet fullfører arbeidsoppgaver oppdateres dette og gjenspeiles i burndown diagrammet. Det er vanlig at x-aksen er dager i sprinten og at y-aksen er resterende arbeid. Burndown diagrammet er et viktig diagram fordi det gir teamet en god forståelse av hvor mye arbeid som gjenstår (Kniberg, 2015).

Et ideelt burndown diagram er en rett linje fra øverste venstre hjørne, til nederst venstre hjørne, men i realiteten ser det ofte ikke slik ut. (Kniberg, 2015)

2.10 Sprint Review – tilbakemelding på leveranser

Sprint Review er en seremoni der utviklerteamet viser hva de har utviklet til produkteiere. Under denne seremonien mottar teamet tilbakemeldinger fra PO, som brukes til å styre utviklerne i retningen av PO sitt visjon for produktet. (Kniberg, 2015, ss. 80, 81)

2.11 Sprint Retrospective – kontinuerlig forbedring

Sprint retrospektiv er en seremoni der teamet reflekterer over hvordan de har gjennomført forrige sprint (Kniberg, 2015, s. 85). Ting som tas opp under en retrospektiv, er hva som gikk bra, hva som gikk dårlig og hvordan teamet kan forbedre seg til neste sprint. Retrospektiven er teamets beste mulighet til å forbedre seg (Kniberg, 2015, s. 84).

Scrumteamnr 10	Versjon: <1.0>
Sluttrapport	Dato: <15/05/25>
dokumentidentifikator	

2.12 Brukertesting

Brukertesting er en metode for å evaluere hvor lett et produkt eller en tjeneste er å bruke, ved å observere hvordan brukere utfører oppgaver. Formålet med brukerstesting er å identifisere problemer i designet, forstå brukeropplevelsen, og finne forbedringsområder (Interaction Design Foundation, u.d.).

For å få de beste resultatene ut av brukertesting er det viktig å planlegge testene godt og definere hva du vil at brukere skal gjøre i løpet av testen. Man bør også lage en rapport av testen slik at man kan analysere brukerens valg og oppførsel sammen med resten av teamet (Interaction Design Foundation, u.d.).

2.13 Universell utforming

Universell utforming handler om å lage og utvikle systemer slik at de kan brukes av alle folk uavhengig av funksjonshemninger. Type hemninger som tas hensyn til er synsproblemer som fargeblindhet eller nedsatt synsevne, motoriske problemer, epilepsi og dysleksi. Det er lovpålagt i Norge å ha en viss grad av universell utforming på IKT løsninger (Rouhani, 2018).

Web Content Accessibility Guidelines er en samling av retningslinjer og krav man kan følge for å sjekke om systemet man utvikler er universell utformet. WCAG gir totalt 61 krav for universell utforming. Lovgivning om universell utforming i Norge sier at offentlig sektor må følge 48 av kravene, mens privat sektor skal følge 35 krav (UUTilsynet, u.d.).

2.14 Personvern og GDPR

General Data Protection Regulation (|GDPR) er en lov satt av EU som stiller krav til behandling av personopplysninger og sikkerhet. GDPR stiller eksempelvis krav til at brukere må gi klart samtykke til at deres data blir behandlet og må vite hva slags data som samles om dem. Reguleringen stiller også krav til at data som behandles både lagres og behandles på en sikker måte, ved for eksempel kryptering av data (Wolford, u.d.).

For å lagre passord på en sikker måte brukes to konsepter: Hashing og salting. En hash-funksjon tar et passord som input og produserer en fast lengde streng, kjent som en hash. En salt er en verdi generert av en kryptografisk sikker funksjon som legges til inputen til hash-funksjonen før hashing, dette gjør hasher unike og mer sikre (Arias, 2025).

For sikker nettbasert autentisert er JWT token og sikre informasjonskapsler vanlig å bruke. Å bruke en «httpOnly» informasjonskapsel gir en sikker lagring av autentiseringstokens på klientsiden (Sola, 2020). Det som lagres i kapselen er et ID-token av typen JWT som er

Scrumteamnr 10	Versjon: <1.0>
Sluttrapport	Dato: <15/05/25>
dokumentidentifikator	

vanlig å bruke for å autentisere brukere og autorisere hva de gjør (auth0, u.d.).

2.15 Sikkerhet

The Open Web Application Security Project (OWASP) Top 10 identifiserer de mest kritiske sikkerhetsrisikoene for moderne webapplikasjoner. To av disse er *A03:2021 – Injection* og *A06:2021 – Vulnerable and Outdated Components*

A03:2021 – Injection omfatter en rekke angrep hvor skadelig data sendes gjennom inputfelter, med mål om å manipulere eller kompromittere systemet. Klassiske eksempler inkluderer SQL-injeksjon, kommandoinjeksjon og Cross-Site Scripting (XSS). Ifølge OWASP er bruk av sikre API-er og Object Relational Mapping Tools (ORMs) blant de mest effektive metodene for å forhindre slike angrep (OWASP, 2021).

A06:2021 – Vulnerable and Outdated Components omfatter bruk av tredjepartsbiblioteker, rammeverk eller andre programvareavhengigheter som har kjente sårbarheter eller ikke lenger vedlikeholdes. For å identifisere og redusere risikoen knyttet til sårbare og utdaterte avhengigheter, anbefales det å benytte verktøy for programvaresammensetningsanalyse, som for eksempel OWASP Dependency-Check. Disse verktøyene skanner prosjektets avhengigheter og sammenligner dem med kjente sårbarheter i databaser som National Vulnerability Database (NVD) (OWASP, 2021).

2.16 Testing

I utvikling av REST-tjenester er testing en viktig del. Fagboken *Spring REST* fokuserer på to hovedtyper: enhetstesting og integrasjonstesting. Java-rammeverk som JUnit og Mockito brukes sammen med Spring Test-modulen og Spring Boot sin starter-test-avhengighet for å forenkle testarbeidet (Varanski & Bartkov, 2021, s. 211-220).

2.16.1 Enhetstesting

Enhetstesting verifiserer at individuelle, isolerte enheter av kode fungerer som forventet. Det er den vanligste formen for testing som utviklere utfører (Varanski & Bartkov, 2021, s. 211). Med bruk av MockMvc og StandaloneSetup kan kontrollere testes uten å starte hele applikasjonen. Dette medfører en muliggjørelse av å simulere http-forespørsler og teste blant annet validering og feilbehandling (Varanski & Bartkov, 2021, s. 216-219).

2.16.2 Integrasjonstesting

Integrasjonstesting har som formål å teste samspillet mellom applikasjonens lag. Ved anvendelse av for eksempel AutoConfigureMockMvc lastes hele Spring-konteksten – dette gir mer realistisk test av REST-endepunktene, og kan gjøre påstander om JSON-respons med verktøy som jsonPath og Hamcrest (Varanski & Bartkov, 2021, s. 220-222).

Scrumteamnr 10	Versjon: <1.0>
Sluttrapport	Dato: <15/05/25>
dokumentidentifikator	

3. Metode

3.1 Scrum

Scrumteamet besto av 7 utviklere og 2 PO-er der en av utviklerne var Scrum master. Det var ingen gruppeleder, men vi valgte heller å ha en flat teamstruktur og ta demokratiske avstemminger om det var uenighet i gruppa. Oppdelingen innad utviklerne var som følger: Backend utviklere: Sander Nessa, Aryan Malekian, Usman Ghafoorzai, Scott du Plessis. Frontend utviklere: Sander Berge, Jonathan Hübertz, Aryan Malekian, Usman Ghafoorzai, Mikael Frøyshov.

For å øke team følelse og kommunikasjon innad teamet dro alle utviklere på bowling sammen en av de første dagene i løpet av prosjektet. Vi tok også lunsj sammen hver dag og diskuterte mange forskjellige temaer som for eksempel politikk.

Det var to PO-er som var master studenter i Digital Transformasjon: Kristian Storehaug og Zakariya Ibrahim. Grunnet manglende engasjement og stor tilbøyelighet blant PO-er bestemte teamet seg å følge relevant litteratur og bruke PO-by-proxy. Hele teamet ble PO-by-proxy og der PO ikke ga klare retningslinjer for hva de ønsket ble disse beslutninger tatt demokratisk av teamet.

3.2 Seremonier

3.2.1 Daily scrum

Ved begynnelsen av hver arbeidsdag kjørte utviklerne daily scrum sammen. I sprint 1 ble dette møtet tatt litt mindre seriøst enn i sprint 2. I sprint 1 satt alle under møtet og vi tok kun opp punktene «hva har jeg gjort siden forrige møte», «hva skal jeg gjøre frem til neste møte» og om det var noe problemer man slet med i arbeidet. Under disse møtene kunne folk lett miste fokus og holde på med noe helt annet, eller diskusjoner rundt ting som ble nevnt kunne starte. For å holde daily scrum til det det behøvde å være startet vi med at personen som snakket sto opp da de snakket og alle andre satt, dette økte terskel for å avbryte, miste fokus eller diskutere andre ting. For å gjøre dette møte mer personlig og empatisk tok hver person også opp hvordan de følte seg og hvordan de hadde sovet. Referent Aryan Malekian tok også notater fra hver daily scrum som ligger i vedleggene [12-22 Møtereferat: 1-12. Scrum Standup].

Scrumteamnr 10	Versjon: <1.0>
Sluttrapport	Dato: <15/05/25>
dokumentidentifikator	

3.2.2 Sprint Planning

Ved prosjektstart og før sprint 2 utførte teamet sprint planning seremonien. Den første sprint planleggingen mente PO de ikke behøvde å delta på, og sa at vi skulle fokusere på å implementere funksjonalitet med høy prioritet i visjonsdokumentet. De ga oss mål om ha wireframes klare til slutten av sprinten. Ettersom PO ikke deltok i møtet og ga veldig løse rammer for prioritet måtte teamet selv bli PO-proxy, og vi tok avgjørelser om prioritering av funksjonalitet innad i teamet. For eksempel prioriterte vi å ha klar logg inn og registreringsfunksjonalitet over admin funksjonalitet, der begge har høy prioritet i visjonsdokumentet. Utviklerne mente at sprint målet om å ha wireframes ferdigstilt var for slapt og bestemte for å sette et eget mål om å ha MVP klar etter første sprint.

Før sprint 2 startet utførte vi også sprint planlegging. I dette møtet var PO delvis med, men måtte forlate møtet etter kort tid. PO-er ga et par punkter på funksjonalitet de ønsket prioritert, men ikke nok til å dekke 2 uker med arbeid og teamet måtte dermed ty til samme løsning som i første sprint planlegging og prioritere og sette sprint mål selv. De ønsket for eksempel at vi skulle prioritere å få implementert gamification i applikasjonen i form av quiz, for referat fra sprint planning 2, se vedlegg[7 sprint planlegging]. Sprint målet som ble satt for sprint 2 var at teamet skulle ha et ferdig produkt og bli ferdig med alle issues som var satt på issue brettet. Dette målet ble satt av teamet ettersom PO var utilgjengelig.

3.2.3 Sprint Review

Sprint review ble utført av Scrum teamet etter sprint 1 og 2. I den første reviewen ga teamet en demo av det som var oppnådd så langt. PO-er var fornøyd og mente vi hadde god progresjon. De mente vi kunne ha en litt mindre kjedelig fargepalett, men at vi kunne være kreative her og finne en vi syntes passet godt. Ellers var det ingen andre tilbakemeldinger enn at de var fornøyde, se vedlegg [10 Sprint 2-review].

Den andre sprint reviewen ble utført siste dagen i sprint 2. Under den reviewen gikk utviklerne gjennom hver user story som var implementert eller prioritert og viste funksjonalitet som tilfredsstilte akseptansekriterier for hver user story. PO-er ga småpirk på noen ting, se vedlegg [11 Sprint 2-Review] for referat fra sprint review 2, men var i all hovedsak veldig tilfreds med resultatet av prosjektet. Med tanke på at tilbakemeldingene ble gitt siste dagen av prosjektet, og hovedprioritet da var å kvitte applikasjonen med så mange bugs som mulig, fikk vi ikke implementert endringene PO ønsket.

3.2.4 Sprint retrospekt

I likhet med sprint review ble retrospektiv utført etter hver sprint. Under første retrospektiv startet vi med at hvert teammedlem beskrev sprinten med et ord, og deretter gikk over til hva folk likte med forrige sprint. Et naturlig tema å ta opp deretter var hva teammedlemmene hadde lært, og hva som kan forbedres til neste sprint.

Scrumteamnr 10	Versjon: <1.0>
Sluttrapport	Dato: <15/05/25>
dokumentidentifikator	

Den andre og siste retrospektiven ble utført litt annerledes. Vi utførte først følelsesregister øvelsen som ble utdelt under Scrum Master møtet for å finne ut om alle var komfortable nok til å dele sine ekte meninger om sprinten. Det første som ble spurt var «Hvis vi skulle gjort sprinten igjen, ville vi gjort det på en annen måte?» og til slutt tok vi opp hva er vi fornøyd med.

3.3 Artefakter

For å støtte planlegging, oppfølging og visualisering av fremdrift i henhold til Scrum-rammeverket, ble det benyttet produktkø, sprintkø og burndown charts. Produktkø ble opprettet og vedlikeholdt i Trello. Dette ble vurdert som hensiktsmessig ettersom det var forventet mange issues, og en fysisk produktkø kunne raskt bli uoversiktlig. Hver kø-oppgave ble markert med estimert innsats og effekt ved bruk av en trello-plugin. Oppgaver markert med høy innsats og lav effekt ble for eksempel nedprioritert for sprint 1. I tillegg ble kortene fargekodet etter prioritet: rød for høy, grønn for middels og gul for lav. Prioriteringene ble delvis gjort i samhold med produkteier under sprint 1 planning, men ettersom begge produkteiere måtte forlate seremonien tidlig ble de fleste prioriteringer avgjort av teamet.

Sprintkø ble administrert fysisk på et issue-board med kolonnene: *Sprint Backlog, To Do, In Progress, Test, Review, Freeze, Fix og Done*. Oppgavene ble skrevet på post-it-lapper og flyttet manuelt mellom kolonnene. Valget om å bruke et fysisk issue-board kom etter en demokratisk avgjørelse innad i teamet, og ettersom teorien beskriver dette som bedre. I sprint 1 ble kun gule post-it-lapper benyttet, og det ble ikke brukt fargekoder for å skille mellom ulike prioriteringsnivåer [3 Sprintkø for sprint 1]. Under sprint planning for sprint 2 ble det besluttet å utvide bruken av farger for å gi en bedre visuell oversikt: gule lapper ble benyttet for oppgaver som ikke ble fullført under sprint 1, oransje for lav prioritet, grønne for middels og rosa for høy prioritet [4 Sprintkø for sprint 2].

Burndown charts ble utarbeidet for hver sprint, og ble brukt til å måle fremdrift. X-aksen representerer dager i sprinten, mens Y-aksen viste antall gjenværende åpne issues. Diagrammene ble lagd for å gi en visuell fremstilling av teamets progresjon mot sprintmålet, og ga en indikasjon på om det ville oppstå forsinkelser eller om arbeidsmengden for sprinten var feilestimert, vedlegg [23 Burndown chart sprint 1, 24 Burndown chart sprint 2].

3.4 Systemarkitektur og design

3.4.1 Overordnet arkitektur

Applikasjonen er organisert i 3 ulike lag.

Det første laget er presentasjonslaget (frontend). Dette laget håndterer alt av brukerinteraksjon og den visuelle fremstillingen. Til dette laget ble Vue.js benyttet.

Det andre laget er tjenestelaget (backend). For forretningslogikk, databehandling og interaksjon med databasen bruker vi Spring Boot. Dette laget gir et REST-API i Java, hvor hvert endepunkt mottar og returnerer data i et fast JSON-format. Frontend og backend

Scrumteamnr 10	Versjon: <1.0>
Sluttrapport	Dato: <15/05/25>
dokumentidentifikator	

kommuniserer kun via disse endepunktene

Det siste laget, databaselaget, er hvor data lagres i en MySQL-relasjonsdatabase. Her brukes Spring Data JPA for å gi et abstraksjonslag mellom tjenestelogikken og SQL-spørringene. Slik kan data skapes, hentes og oppdateres uten å skrive ren SQL.

3.4.2 Teknologier brukt

Spring Boot (Java) kompiles og pakkes med Maven, som også håndterer avhengigheter, bygg og tester. Alle REST-endepunkter er dokumentert med Swagger, som gir god oversikt over backend endepunktene. Spring Security ble benyttet for sikkerhet med autentisering og autorisasjon med roller, samt JWT (JSON Web Token) som lagres i HTTP-only cookie, slik at tokenet er utilgjengelig for klient-skript. Tokenet følger hver forespørsel og gir statløs innlogging. Alt av produktkategorisering og andre algoritmer kjøres på serversiden for å avlaste klientsiden (legg til spesifikke algoritmer). All kode er skrevet i TypeScript med Vite for rask bygging og reload. All kode er skrevet i TypeScript. De samme DTO-definisjonene brukes på backend og frontend for å sikre type-konsistens. Pinia (også i TypeScript) lagrer applikasjonsdata som global state. Axios ble brukt for å sende HTTPforespørsler til Springserveren.

3.5 Testing

3.5.1 Brukertesting

Det ble utført totalt 2 brukertester i løpet av utviklingen. Den første brukertesten ble utført med produktet som var regnet som MVP, altså det vi hadde klart å få utviklet i løpet av sprint 1. Brukertest 1 ble utført av Fredrik Fahlstrøm med utvikler Scott du Plessis som testobservatør. Fahlstrøm er en 20 år gammel mann som studerer økonomi og administrasjon og er interessert i beredskap, men skulle ønske han visste mer om beredskap og er dermed en god kandidat for brukertest av programmet. Før brukertesten ble utført forklarte testobservatør hva målet med testen er og at ingenting brukeren gjør er feil, men at testen er en evaluering av design. Oppgavene som ble utført var forhåndsdefinert med suksesskriterier. Observasjoner og kommentarer ble skrevet ned fortløpende i løpet av testen. For en detaljert beskrivelse av brukertest 1 se vedlegg [5.1 Brukertester].

Den andre brukertesten ble utført mot slutten av sprint 2 da produktet begynte å bli ganske ferdig. Brukertest 2 ble utført av Linus Backen, en 21 år gammel mann som studerer nanoteknologi. Han vet ikke mye om beredskap, men etter spørsmål om han ønsker bedre kunnskap om beredskap var svaret utelukkende ja, og dermed var en god kandidat for brukertest. Mange av de samme oppgavene som ble utført i brukertest 1 ble utført i test 2 også, for å se om Backen hadde noen andre innvendinger enn Fahlstrøm. I tillegg ble all ny funksjonalitet testet som hadde blitt lagt til etter brukertest 1. Utvikleren Jonathan Hübertz var testobservatør under test 2 og fulgte samme metode for brukertesten som ble gjort i test 1, for en mer detaljert beskrivelse av brukertest 2 se vedlegg [5.1 Brukertester].

Scrumteamnr 10	Versjon: <1.0>
Sluttrapport	Dato: <15/05/25>
dokumentidentifikator	

Observasjonene som ble gjort under brukertestene ble diskutert i plenum med hele utviklerteamet og en plan for å fikse de avdekkede feilene ble laget.

3.5.2 Automatisk testing av programvare

I løpet av utviklingen av produktet ble automatiske tester laget. Enhetstester ble laget for service klassene i backend, her ble JUnit, spring test og Mockito brukt for enhetstesting. I frontend lagde vi også enhetstester under utvikling og brukte Vitest for å lage tester. Ved enhver tid i utviklingen forsøkte vi å dekke kravene for test dekning av kode.

Mot slutten av prosessen utviklet vi integrasjonstester for å sjekke at de utviklede komponentene fungerte sammen som vi antok de skulle. For å sjekke at alle lag i backend (kontroller, service, database) fungerte sammen som de skulle lagde vi integrasjonstester med Mockmvc og Spring Boot test. For å sjekke at hele applikasjonen fungerte som den skulle brukte vi Cypress for å lage e2e tester.

3.6 Personvern, GDPR og sikkerhet

3.6.1 GDPR og personvern

For at nettsiden skulle være i så godt som mulig samsvar med GDPR så lagde teamet en personvernerklæring som er lett tilgjengelig på nettsiden. Når en person registrerer en bruker er det et krav om at de godkjenner personvernerklæringen, altså de får ikke registrert uten å ha gitt godkjenning og samtykke til vår behandling av deres data. Dette for å samsvare med GDPR.

I tillegg til personvernerklæringen lagres sensitive data om brukere på en sikker måte. Passordene til brukerne blir saltet og kryptert med en sikker algoritme før de lagres i databasen. For sikker autorisasjon og autentisering ble JSON Web Tokens brukt, og de ble lagret som httpOnly informasjonskapsler på klientsiden.

3.6.2 Sikkerhet

For å oppfylle kravene til sikkerhet slikt spesifisert i visjonsdokumentet, ble det iverksatt konkrete tiltak med utgangspunkt i OWASP Top 10. Minstekravet var å adressere kategorien A03:2021 – Injection. I tillegg ble tiltak rettet mot A06:2021 – Vulnerable and Outdated Components implementert for å styrke sikkerheten.

For å redusere risikoen for injeksjonsangrep (A03), ble det benyttet en Object Relation Mapper (ORM) gjennom Java Persistence API (JPA). Dette er i tråd med OWASP sine anbefalinger for hvordan injeksjonsrisiko skal best håndteres. Grunnen til at teamet valgte denne løsningen er fordi det i tråd med OWASP sine anbefalinger, samt at flere teammedlemmer hadde forkunnskaper om JPA.

Videre ble det implementert en automatisk sikkerhetsskanning av prosjektets avhengigheter

Scrumteamnr 10	Versjon: <1.0>
Sluttrapport	Dato: <15/05/25>
dokumentidentifikator	

ved hjelp av OWASP Dependency-Check. Verktøyet ble satt opp i en egen gren av CI/CD-pipelinen for å identifisere kjente sårbarheter i tredjepartsbiblioteker, som ledd i å adresse A06. Ettersom nedlasting av NVD-basen viste seg å være ressurskrevende – og ingen løsning på caching ble funnet – ble analysen kjørt på en separat gren fra hvor hovedutviklingen av applikasjonen foregikk. Tilgang til NVD med API-nøkkel ble etablert for å haste opp prosessen. Den genererte sårbarhetsrapporten blir lastet opp som et artefakt til Gitlab.

Under første gjennomføring av sjekken ble det avdekket 5 forskjellige sårbarhetsrisikoer i vedlegg [25 OWASP dependency check]. Fire av disse med en høy alvorlighetsgrad, og en med middels.

3.7 Universell utforming

3.7.1 WCAG og WAVE

Under utvikling har teamet tatt hensyn til WCAG krav og universell utforming. Et verktøy som ble brukt som hjelp med å designe universelt er «WAVE evaluation tool». Ved å bruke dette verktøyet på nettsiden fikk man varslere om for dårlige kontraster mellom farger, verktøyet sjekket også HTML-koden for å se at alt var navigerbart med tastatur og at alle relevante elementer har alt-tekst. Eksempler på ting som måtte endres var rød tekst med grønn bakgrunn, noe som gir dårlig utforming for personer med rødgrønn fargeblindhet, vi måtte legge til ARIA-knagger på noen elementer i HTML-koden for å gjøre dem navigerbare og integrerbare med tastaturet.

Etter en grundig gjennomgang av nettsiden med WAVE ble WCAG-matrisen til UUTilsynet gjennomgått og hvert krav ble sjekket på siden. Ferdig utfylt WCAG-matrise ligger som vedlegg [6 WCAG matrise].

3.7.2 CI/CD

Hver gang noen pushet kode til GitLab kjørte en pipeline. Pipelinene på både frontend og backend bygget og kjørte enhetstester. Hvis pipelinen feilet, kunne man ikke merge koden sin inn i dev eller main branch. Backend pipelinen hentet autogenerated swagger dokumentasjon av endepunktene i spring Rest api-et vårt, i tillegg til å hente jacoco test deknings rapport, og deployet begge disse til GitLab pages. Frontend pipelinen hentet autogenerated test dekning rapport laget av istanbul code coverage og deployet dette på GitLab pages.

Vi hadde en egen branch på GitLab som hadde en egen pipeline som sjekket OWASP dependency check som tidligere beskrevet under 3.3.2 Sikkerhet.

Scrumteamnr 10	Versjon: <1.0>
Sluttrapport	Dato: <15/05/25>
dokumentidentifikator	

4. Resultater

4.1 Prosjektresultater

4.1.1 Måloppnåelse Sprint 1

Sprint målet for sprint 1 var å legge fundamentet for applikasjonen og ha en simpel MVP å presentere produkteierne med. Dette kan sies å ha blitt oppfylt selv om ikke alle de utvalgte brukerhistoriene ble oppfylt. Ettersom PO-ene ikke kom med noen tydelige prioriteringer for hva de ønsket oppnådd etter sprinten gikk teamet i gang med å implementere funksjonaliteten vi anså som viktigst. Dette innebar blant annet innlogging, registrering, kart og beredskapslager. Se [26 brukerhistorier sprint 1] for en omfattende oversikt over sluttstatus av alle brukerhistorier som ble satt som mål å utføres i sprint 1. Brukerhistoriene som ikke ble fullført ble overført videre til sprint 2.

Under sprint review for sprint 1 kom inntrykk frem at PO-ene hadde et godt inntrykk av applikasjonen. PO-ene ga beskjed om å prioritere gamification gjennom en beredskapsscore samt quiz. Et annet ønske var at nettsiden skulle tilby informasjon som motiverte ikke-registrerte brukere til å registrere seg. Ellers var tilbakemeldingene litt vage: teamet fikk beskjed om å selv bestemme design og layout på nettsiden, og presiserte ingen preferanse i noen retning når teamet foreslo ulike fargepaletter.

4.1.2 Måloppnåelse Sprint 2

Målet med sprint 2 var å komme i mål med alle gjenstående brukerhistorier fra sprint 1, i tillegg til alle nye brukerhistorier som kom frem under sprint planning. Dette ville sikre at vi kunne levere en ferdigstilt applikasjon som var i tråd med både P.O. - og PO-by-Proxy – sine krav til prosjektet.

Alle de gjenværende brukerhistoriene fra sprint 1, samt nye brukerhistorier til sprint 2, ble fullført under denne sprinten. Enkelte akseptansekriterier knyttet til de ulike brukerhistoriene ble utelatt - enten grunnet mangel på tid, eller etter avklaring med PO eller PO-by-Proxy, se vedlegg [28 ikke fullførte brukerhistorier].

PO sine tilbakemeldinger under sprint review var positive. Det ble uttrykt stor tilfredshet med teamets innsats og fremdrift under hele sprinten, og teamet fikk klarsignal til å levere applikasjonen.

4.1.3 Tilbakemeldinger på brukertester

I brukertest 1 fikk teamet verdifulle tilbakemeldinger fra en representativ bruker. På registreringssiden fikk vi tilbakemelding om at dersom et passord ikke oppfyller sikkerhetskravene og brukeren forsøker å endre det til et nytt passord som heller ikke oppfyller kravene, bør feilmeldingen oppdateres for å gjenspeile den nye valideringen, slik at bruker skjønner at det nye passordet fortsatt ikke samsvarer med sikkerhetskravene. I tillegg til dette fikk vi tilbakemelding på at før, under og etter boksene på hovedsiden ikke

Scrumteamnr 10	Versjon: <1.0>
Sluttrapport	Dato: <15/05/25>
dokumentidentifikator	

gir gode forklaringer på hva man finner i de nevnte undermenyene. På lageret fikk vi tilbakemelding om at hele boksen for en vare burde være redigerbar hvis antall av noe skal endres på, istedenfor å ha en rediger knapp på hver vare. På kartet fikk vi tilbakemelding om at ikonet for tilfluktsrom kunne være vanskelig å se på grunn av fargene på ikonet i forhold til bakgrunnen. Ettersom at det var forholdsvis enkle feil ble aktuelle endringer implementert raskt for å få et mer brukervennlig design.

I brukertest 2 fikk vi også verdifulle tilbakemeldinger på designet. Eksempel på tilbakemelding var på «Sjekk epost» siden, der hadde vi flere knapper som «åpne outlook» som ikke fungerte, dette fant bruker ut av da han skulle klikke på de knappene. Vi fikk også beskjed om at infotekst om hva man fant på hver av sidene før, under og etter var litt liten og vanskelig å se. På samme måte som i brukertest 1 var det ingen store feil, men relativt enkle feil å fikse og ble fikset fort.

For mer detaljert beskrivelse av brukertest 1 og 2 se vedlegg [5.1 Brukertester].

4.2 Overordnet måloppnåelse i forhold til visjonsdokumentet

4.2.1.1 Funksjonelle Krav

Når det kommer til de funksjonelle kravene i visjonsdokumentet, ble de fleste brukerhistoriene oppnådd. I vedlegg [28 ikke fullførte brukerhistorier] kan man se en oversikt over alle brukerhistoriene som ikke ble oppnådd i noen grad, samt beskrivelser på hvorfor de ble utelatt.

4.2.1.2 Ikke-Funksjonelle Krav

De ikke-funksjonelle kravene beskrevet i visjonsdokumentet ble også oppfylt: Dekning av kode fra programvaretester i backend ligger på 57%. En god blanding av integrasjonstester og enhetstester er brukt for å oppnå dette. I frontend ligger dekningsgraden av enhetstester rett i underkant av 30%, men cypress e2e tester er også laget, men dekningsgraden deres telles ikke i dekningsrapporten på GitLab pages, total dekning på frontend er da over 30%. For detaljerte test-rapporter av både frontend og backend, sjekk GitLab pages eller se vedlegg [5 GitLab wiki, test].

Når det kommer til applikasjonens sikkerhet, er det implementert både autentisering og autorisering. Dette er beskrevet i mer detalj tidligere i rapporten. OWASP A03 beskyttes ved bruk av JPA som beskytter mot SQL-injeksjoner, Vue.js, axios og Spring Boot beskytter mot andre XSS svakheter. OWASP A06 beskyttes ved bruk av OWASP Dependency-Check, dette er mer beskrevet i [METODE: SIKKERHET].

Resultat av utfylling av WCAG-matrisen til UUTilsynet ga 27 ut av 61 krav dekket. Alle krav på WCAG prinsipp 2.1 ble sjekket og ble enten godkjent eller bestemt at vi ikke hadde den type innhold på siden.

Scrumteamnr 10	Versjon: <1.0>
Sluttrapport	Dato: <15/05/25>
dokumentidentifikator	

4.3 Administrative resultater

4.3.1 Sprint Retrospektiv

Retrospektivene for sprint 1 og 2 viser både læringspunkter og forbedringer i teamets arbeidsprosess. Under sprint 1 møtte teamet på utfordringer tilknyttet store og uoversiktlige issues. Det manglet også testing på frontend – samt lite strukturert prosjektoversikt og svak forståelse av visjonsdokumentet. Følgelig resulterte dette i svak progresjon, noe som gjenspeiles i burndown-charten fra sprint 1 [23 Burndown chart sprint 1]. Grafen viser mange åpne issues mot sprint-utløpsdato og stort avvik fra den ideelle grafen. Under *daily stand-up* satt teammedlemmene (istedenfor å stå oppreist), og konsekvent var det tendenser til lite engasjement og energi. Tross dette klarte teamet å levere en MVP. Teamet opplevde fremgang og trivsel, med en flat struktur og godt samspill. Disse erfaringene førte til flere konkrete forbedringer i sprint 2. Deriblant fargekoding for prioritering ble innført, og antallet og størrelsen på issues ble redusert. En annen forbedring var at stand-up var gjennomført stående – med fokus på søvn og status. Teamet opplevde at dette økte engasjementet – og det var lettere å forholde seg til hverandre dersom noen hadde en dårlig dag. Det ble også økt bevissthet rundt dokumentasjon, samarbeid og ansvarsfordeling. Refleksjoner fra enkeltmedlemmer tyder på at prosjektet er vært lærerikt, men at det var behov for bedre planlegging, deling av kunnskap og tydeligere kommunikasjon – internt og spesielt med PO.

4.3.2 Samarbeid og teamutvikling

I starten av prosjektet kjente ikke alle hverandre i teamet, og som et resultat var det ekstra viktig å styrke det sosiale. Tiltak som felles lunsjer hver dag og bowling, bidro til at teamet ble bedre kjent og samholdet ble styrket. I tillegg hadde teamet daglige standups, som bidro til jevn og tydelig kommunikasjon. Dette medførte til at kommunikasjon og samarbeid ble bedre utover prosjektet. Vi ble også flinkere på å ta hensyn til hverandres arbeidsstil, noe som er ulikt blant alle i teamet, som ga oss bedre effektivitet utover prosjektet.

4.3.3 Roller og ansvar

Teamet var organisert med en flat teamstruktur, der alle hadde lik stemmerett på avgjørelser og arbeidet ble fordelt likt. Dette ga et mer inkluderende miljø, men skapte utfordringer når det kom til beslutningstaking. I noen tilfeller tok det unødvendig lang tid å lande avgjørelser ettersom PO ikke ga tydelige svar. Utydelige og trege svar fra PO var en av de største (gjengående) utfordringene i løpet av prosjektet. Dette førte til at PO-by-Proxy ble iverksatt og teamet tok avgjørelser basert på antakelser om hva PO ville ønsket.

Scrumteamnr 10	Versjon: <1.0>
Sluttrapport	Dato: <15/05/25>
dokumentidentifikator	

4.3.4 Artefakter

Den første sprint backloggen vi utarbeidet etter første sprint planning, bar preg av at vi hadde litt for høye forhåpninger om hva vi kunne få til i løpet av sprinten. Som man kan se på burndown-charten fra sprint 1, vedlegg [23 Burndown chart sprint 1], klarte vi ikke å bli ferdig med alle issues vi lagde. Dette var noe vi tok i betraktning til neste sprint planning, altså å ikke planlegge altfor mye arbeid. En annen ting vi lærte om sprintkøen etter sprint 1 var at vi hadde delt opp i altfor store issues.

Dette var noe vi tok i betraktning under neste sprint planning. Ut ifra burndown-chart fra sprint 2, vedlegg [24 burndown chart sprint 2], kan man se at tiltakene vi iverksatte fungerte fint for å få en mer jevn nedgang i antall issues, og vi klarte å planlegge ganske greit i forhold til hvor mye arbeid vi la opp til i den sprinten.

5. Konklusjon og videre arbeid

5.1 Oppsummering av prosjektets sluttresultat

Prosjektet må anses som vellykket, både med tanke på sluttproduktet og hvordan det ble utviklet. Krisefikser ble levert med høy grad av funksjonalitet, og de viktigste kravene i visjonsdokumentet ble oppfylt. Dette inkluderer funksjoner som innlogging, registrering, lageroversikt og kartvisning. I tillegg ble det utviklet en quiz som del av gamification-kravene. Brukertestene bekreftet at løsningen var brukervennlig og relevant. Samlet sett viser prosjektet høy grad av kravoppfyllelse, teknisk kvalitet og god prosess etter justeringer. Resultatet er et sikkert og funksjonelt produkt som dekker behovene definert i prosjektets visjonsdokument. Vi har oppfylt alle krav/ønsker fra produkteiere, og teamet selv er fornøyd med det endelige resultatet.

5.2 Evaluering av utviklingsprosess og metode

Å benytte Scrum og smidig utvikling som rammeverk for prosjektet har i stor grad fungert godt. Teamet mener at iterativ utvikling i form av sprinter, daily stand-up og sprintretrospektiv har bidratt til struktur, kontinuerlig fremgang og forbedring. Teamet kan konkludere med at det har vært veldig nyttig med korte feedbacksløyfer for å oppdage forbedringspunkter og justere arbeidsprosessen fortløpende – mest internt. Bruken av fysiske artefakter, som et fysisk scrum-board, kombinert med digital produktkø i Trello, ga en visuell og oversiktlig fremstilling av fremdriften. Dette gjenspeiler i overgangen fra sprint 1 til sprint 2, der tydelig effekt av smidig tenkning vises. Dette innebærer tiltak som fargekoding av oppgaver, mindre og mer spesifikke issues, og stående stand-ups som førte til økt engasjement og mer effektiv arbeidsflyt. Retrospektivene var en vital praksis i prosjektet, ettersom det i sprint 1 ble avdekket flere svakheter, som uklare issues. Tiltakene som ble besluttet under retrospektiven hadde positive effekt i sprint 2.

Det er verdt å nevne at teamet erfarte flere svakheter knyttet til utføringen av Scrum. En

Scrumteamnr 10	Versjon: <1.0>
Sluttrapport	Dato: <15/05/25>
dokumentidentifikator	

utfordring var manglende tilgjengelighet og tydelighet fra PO. Dette resulterte i at teamet tok i bruk en PO-by-proxy-modell. Her tok hele teamet ansvar for prioritering og beslutninger i de situasjoner der PO-ene var fraværende eller ikke ga klare signaler. Selv om dette ga en noe fremdrift, førte det også til tidkrevende diskusjoner og unødvendig usikkerhet. I etterkant kan teamet konkludere med at det ville vært mer effektivt å utnevne én person som PO-proxy.

Valget av å ha en flat teamstruktur resulterte i inkludering og bred involvering der alle fikk en stemme. Samtidig oppstod det enkelte ganger utfordringer med beslutningstaking, som for eksempel ved valg av design, der teamet var uenig om fargevalg og design av sidene. Dermed kan det konkluderes med at selv om flat struktur kan fungere godt, bør det suppleres med tydelig rollefordeling i situasjoner som krever effektiv ledelse eller avklaring.

5.3 Refleksjon over teknologisk valg

Typescript, og gjenbruk av DTO-er mellom backend og frontend, ga typesikkerhet, enklere endring av dataformatet som sendes og reduserte feil knyttet til dataformat. GitLab CI med automatiske tester og OWASP Dependency Check ga tryggere leveranser og synlige sårbarheter. Enkelte designvalg i frontend var vanskelige å bli enige om, og universell utforming krevde ekstra innsats og læring.

Teamet lærte hvordan CI, sikkerhet og testing bør integreres tidlig i utviklingsløpet. Bruk av verktøy som WAVE ga økt bevissthet om tilgjengelighet. Underveis i prosjektet fikk teamet mulighet til å bli bedre kjent med prinsipper om universell utforming og implementasjonen av disse.

5.4 Læringsutbytte og teamutvikling

De sosiale tiltakene som ble tatt ga veldig positiv påvirkning på trivsel. Når det kommer til effektivitet kan det diskuteres for at det gikk mye i skravling og tulling ettersom at teamet ble godt kjent med hverandre. Likevel ble målene nådd, og gruppa er fornøyd med produktet.

Teamet tar med lærdom fra prosjektet. Teamet har blitt rutinerte med scrum og alle de tilhørende seremoniene, og kan føle seg mer klare for arbeidslivet med dette prosjektet i sin portefølje. Faglig sett har teamet lært masse nytt, for eksempel hvordan å bruke websocket for real-time oppdateringer, bruke JPA istedenfor å skrive egne SQL spørringer og i tillegg å designe en universelt utformet nettside ved bruk av UUTilsynets matrise.

5.5 Forbedringspunkter og anbefalinger

Med en ekstra sprint ville teamet forbedret dokumentasjon og utnevnt én bestemt PO-proxy, og testet wireframes. For videre utvikling anbefales tydeligere rollefordeling, mer automatisert testing og bedre CI-oppfølgning. De som overtar prosjektet bør sikre tett dialog

Scrumteamnr 10	Versjon: <1.0>
Sluttrapport	Dato: <15/05/25>
dokumentidentifikator	

med PO, bruke små og hyppige pull requests, og holde retrospektiver fast. Dokumentasjon bør være oppdatert, lett tilgjengelig med både fysisk og digital oppgaveoversikt med tydelig prioritet.

Bibliografi

- Arias, D. (2025, Januar 17). *auth0.com*. Hentet fra Adding Salt to Hashing: A Better Way to Store Passwords: <https://auth0.com/blog/adding-salt-to-hashing-a-better-way-to-store-passwords/>
- auth0. (u.d.). *auth0.com*. Hentet fra JSON Web Tokens: <https://auth0.com/docs/secure/tokens/json-web-tokens>
- Deemer, P., Benefield, G., Larman Craig, & Vodde, B. (2009). *The Scrum Primer*.
- GeeksforGeeks. (2023, Februar 9). *GeeksforGeeks.no*. Hentet fra Introduction of object-oriented programming: <https://www.geeksforgeeks.org/introduction-of-object-oriented-programming/>
- Interaction Design Foundation. (u.d.). *Interaction Design Foundation*. Hentet fra Usability Testing: https://www.interaction-design.org/literature/topics/usability-testing?srltid=AfmBOor8F1_vooxsEM9UsZdaeAxBAszeRkR942wYfKxFHIT3175upgeb
- Kniberg, H. (2015). *Scrum and XP from the trenches, How we do scrum*. Toronto: C4Media.
- OWASP. (2021). Hentet fra A03:2021 – Injection: https://owasp.org/Top10/A03_2021-Injection/
- OWASP. (2021). Hentet fra A06:2021 – Vulnerable and Outdated Components: https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/
- Rouhani, M. (2018). TDAT2003 Systemutvikling 2 med web-applikasjoner, Menneske-Maskin Interaksjon – Universell Utforming . Trondheim, Trøndelag, Norge.
- Sola, P. (2020, august 22). *dev.to*. Hentet fra Today's rabbit hole: securing JWTs for authentication, httpOnly cookies, CSRF tokens, secrets & more: <https://dev.to/petrussola/today-s-rabbit-hole-jwts-in-httponly-cookies-csrf-tokens-secrets-more-1jbp>
- UUTilsynet. (u.d.). *UUTilsynet.no*. Hentet fra WCAG-standard: <https://www.uutilsynet.no/wcag-standard/wcag-standard/86>
- Varanasi, B. &. (2022). *Spring REST: Building Java microservices and cloud applications [PDF]*. Apress.
- Vihovde, E. H. (2025, Mai 12). *snl.no*. Hentet fra Store norske leksikon: https://snl.no/objektorientert_programmering
- Wolford, B. (u.d.). *gdpr.eu*. Hentet fra What is GDPR, the EU's new data protection law?: <https://gdpr.eu/what-is-gdpr/?cn-reloaded=1>