Hopla prosjektplan

Bjørn Tore Lødding
Ane Marie Øglænd Johnsen
Vilde Adela Svejkovksy Kværne

Januar 2025

Innhold

Kapittel 1: Mål og Rammer

1.1 Bakgrunn

Hopla er et nystartet privat firma som ble grunnlagt for å gjøre rytterens hverdag enklere. [1] I 2022 var det om lag 100 000 hester i Norge, men det er fremdeles ingen oversikt over hvor man har lov til å ferdes i naturen med hest [2]. Informasjon om hva slags føre det er på turstiene samt oppdateringer om tilstanden er svært vanskelig å skaffe seg. Per dags dato finnes det ikke en samlet plattform for denne informasjonen, som har ført til at den midlertidige løsningen har blitt at hesteeiere bruker mange ulike Facebook-grupper for å høre om hvor man kan ri, som igjen er uoversiktlig.

Om du skal ri på lengre turer, er det nyttig å vite om det er parkeringsplasser i nærheten. Vanskelighetsgraden på turene, egnet årstid for å ri og kjøre med hest samt krevende partier i løypen er nyttig informasjon å vite for hestefolk. Denne informasjonen ønsker Hopla å gjøre tilgjengelig for alle hesteinteresserte i Norge, ved å samle alt på ett sted - rett i lommen.

1.2 Prosjektmål

Vi har valgt å dele målene for prosjektet vårt inn i 3 deler: resultatmål, effektmål og læringsmål. Resultatmål er konkrete, målbare utfall av prosjektet. Effektmål er de langsiktige endringene som vi ønsker å oppnå med prosjektets resultater, og læringsmål er hva studentene i prosjektet skal lære og utvikle av kunnskaper gjennom prosjektet.

Resultatmål
Målet er å utvikle en applikasjon for de mest brukte plattformene (Android og iOS) [3] slik at ryttere har informasjon om ruter rett i lommen, og i tillegg et webgrensesnitt som kan administrere databasene. Alle grensesnittene skal kunne gi statistikker eller grafer om nåværende og fremtidig utvikling av brukerbasen. Appen skal kunne legges ut på App Store og Google Play av oppdragsgiver uten behov for modifikasjoner.

Effektmål

- Lansere en fullverdig tur-applikasjon for ryttere som viser forskjellige turer i Norge både i et kart- og listeformat.
- Hopla skal være rytteres primære tur-applikasjon i Norge med all relevant informasjon samlet på ett sted.
- Bedre oversikt over tilgjengelige turløyper for rytter og hest uavhengig av ferdighetsnivå.
- Raskere og bedre informasjon om tilstanden på turløypene oppdatert av appens brukere.
- Samle all informasjon rundt rideturer på ett sted.

- Applikasjonen skal hjelpe ryttere med å være sikker på ridetur ved å samle informasjon fra ryttere som går samme turen
- Gi administrasjonen til appen statistikk fra brukere på appen for å kunne gi best mulig brukeropplevelse

Læringsmål
Teamet ønsker å lære hvordan man jobber sammen i et team, men vi skal også dypere inn i følgende temaer og verktøy:

- Bruk av Overleaf som et rapportskrivings verktøy.
- Bruk av rammeverket .NET.
- Nytt programmeringsspråk: C#.
- Integrasjon mellom mobilapp og databaser.
- Kartfunksjoner og plassering av symboler.
- Sortere ut og kun vise det som er med i kartutsnittet.
- Sikker behandling av sensitiv informasjon.
- Jobbe med SCRUM i et større prosjekt.

1.3 Rammer

- iOS applikasjonen skal fungere for alle telefoner fra iOS 13 [4].
- Android applikasjonen skal fungere or alle telefoner med API level 16 [5].
- Webgrensesnittet skal fungere for nettlesere som støtter HTML5 [6].
- Databasen skal settes opp slik at de lett kan oppgraderes av oppdragsgiver i senere tid.
- Koden skal bygges slik at den lett kan pakkes sammen og legges inn som app på App Store og Google Play etter prosjektets slutt, om dette er ønskelig for oppdragsgiver.

Kapittel 2: Omfang

2.1 Fagområde

Hopla ønsker at programvaren skal utvikles til iOS og Android, og at man skal kunne laste ned appen fra App Store og/eller Google Play. I tillegg ønsker de en nettside som er forbeholdt dem, der de kan administrere portalen.

Hopla hadde ingen krav når det kom til valg av teknologier, og vi fikk derfor en stor frihet til å velge selv. For å komme fram til en liste med valgte teknologier var det flere valg som måtte vurderes, blant annet valg av løsning for plattformutvikling til frontend og backend samt hvilke språk som skal brukes. I tillegg måtte vi finne ut hva som var den beste løsningen når det gjelder databasemodell og backend-språk i forhold til dette.

Vi har vært innom og vurdert flere teknologier og verktøy vi har erfaring fra i tidligere emner, men valgte også å vurdere alternativer vi ikke har vært borti før. Både for å kunne få større læringsutbytte av prosjektet, men også for å finne det som virker teoretisk best for prosjektet vårt ut ifra undersøkelser vi har gjort. Vi ønsket også å være åpne for alle muligheter og ikke nødvendigvis bruke et språk fra et tidligere fag fordi vi liker eller kjenner det godt. Vi ønsket å vektlegge følgende kriterier når vi gjorde beslutningene angående en best mulig teknologistakk:

- Fremtidsrettet og relevant i arbeidslivet
- Rask responstid, effektivitet og ytelse
- Skalerbarhet
- God integrasjon mellom frontend, backend og SQL-databaser
- Sikkerhet
- Stabilitet
- Støttes av populære hosting-tjenester
- God integrasjon med mobilens funksjonaliteter

Vi har undersøkt 4 forskjellige valg vi har når det gjelder plattformer : Native, Crossplatform, Crossplatform hybrid eller PWA (Progressive web apps)[7]. Hvis vi velger å gå for en crossplatform-løsning, så skriver vi programmet med samme språk til både iOS og Android, som er en stor fordel da vi slipper å skrive egen kode for hver av dem. På en annen side må vi skrive flere unntak i koden av typen "if Android -¿ do this/if iOS -¿ do this". Dette fordi de har forskjellige måter å operere innebygde funksjoner, slik at det allikevel blir en del dobbel koding. Koden vil være universell for begge plattformer, men vil bære litt preg av "one size to fit them all". Dette betyr også at når kodebasen i seg selv blir større, blir det også mer batterikrevende for telefoner, noe vi ønsket å minimere. Det siste punktet er bruk av innebygde funksjoner på mobiler som blant annet GPS til kartet, som kan være mer komplisert på en crossplatform-løsning.

Det vi har funnet ut med en crossplatform-løsning er at noe funksjonalitet blir dårligere og at ytelsen kan være redusert om vi velger ett design som skal gjelde for både iOS og Android. Siden dette er en bacheloroppgave i programmering, ønsker vi ikke å inngå slike kompromisser og velger derfor å ikke bruke crossplatform-løsningen. Når det gjelder PWA, har vi ingen tidligere erfaring med løsningen. Etter å ha undersøkt alternativet, lærte vi at det er begrensninger for hvor mye og hvor kompleks funksjonaliteten vi legger til kan være [8]. Dette kan gå på kompromiss med flere av base funskjonalitetene til applikasjonen. Fordi vi ikke er sikre på hvordan dette vil påvirke vår programvare, har vi derfor bestemt oss for å gå for en Native-løsning og kommer til å bruke Swift for iOS og Kotlin for Android.

Fordelene med Native-løsning er at kodebasen blir mindre som fører til lavere strømforbruk på mobil, lettere tilgang til mobilens funksjonaliteter som er viktig på grunn av gps sin rolle i applikasjonen, og til slutt bedre app prestasjon [7]

Administrasjonen av nettsiden skal gjøres med en nettside, som kobler til en bestemt adresse, f.eks admin.hopla.no. Her kan vi velge mellom HTML/JavaScript eller at backend genererer nettsidene. Siden dette skal gjøres av Hopla-administrasjonen, så er det ikke kritisk at backend står for denne jobben. Hvis backend skulle generert HTML til andre brukere, så ville det vært mer sårbart ved et angrep av typen generering av enorme data fra flere brukere som kunne overbelastet serveren.

Når det gjelder valg av språk til backend er det flere å velge mellom, blant annet JavaScript/TypeScript, Python, Java, Rust og .NET. Flere av disse er gode alternativer og oppfyller kriteriene som beskrevet i begynnelsen av dette kapittelet. JavaScript/TypeScript har vi en del erfaring med, samt Python og bruk av Flask for å lage nettbutikk. Ved undersøkelser av de ulike alternativene fant vi ut at Python kan være tregere for sanntids-oppdateringer, mens JavaScript er tregere ved CPU bruk. .NET skal være det språket med best ytelse, men er et "tyngre" språk å bruke med tanke på læringskurve og oppsett. Siden vi også ønsker å lære mest mulig og at .NET oppfyller kriteriene svært godt, så er det dette vi endte opp med.

Til slutt må vi velge databasemodell. Ut i fra kravene som er beskrevet i oppgavebeskrivelsen (2.3), så vi på disse som et mer komplisert databasesystem, og dermed bestemte vi oss for en SQL-database. Det virket som at de fleste leverandører av nettjenester støtter PostgreSQL, så derfor synes vi at dette er det beste valget. [9] Da vil ikke Hopla være bundet til én leverandør, og det vil ikke være databasedelen som begrenser videre utvikling, hvis man i fremtiden ønsker å bytte leverandør. Vi har også muligheten til å bruke PostGIS, som er en utvidelse av PostgreSQL som gjør det mye enklere å blant annet beregne avstander som å finne nærmeste rute, da dette kan være svært ressurskrevende å beregne mellom nåværende posisjon mot hvert punkt i ruten.

Dette førte til disse valgene innenfor programmeringsspråk og teknologier, som kan sees i figur 5.1:

- Swift for iOS-utvikling på grunn av optimalisering rettet spesifikt mot iOS [10].

- Kotlin for Android-utvikling på grunn av dens moderne UI toolkit og optimalisering for spesifikt Android [11].
- HTML/JavaScript/CSS for nettsiden grunnet samhandlingen deres for å skape et intuitivt og pent grensesnitt, samt dems gode røtter i utviklingsmiljøet for frontend utvikling [12].
- .NET for RESTful API på grunn av dens gode sikkerhet, robusthet og prestasjon selv om .NET har en vanskelig læringskurve [13].
- SQL for databasen grunnet samhandlingen mellom de ulike delene av databasen som er nødvendig [14].

For hovedinnholdet av vår bacheloroppgave, vil vi spesielt ta i bruk kunnskap fra følgende emner fra tidligere i bachelorløpet våres:

- **PROG1004** Programvareutvikling.
- **IDATG2204** Datamodellering og Databasedesign.
- **PROG2053** Webteknologier.
- **PROG2005** Cloud Technologies.
- **PROG2007** Mobilprogrammering.
- **IIKG2001** Software Security.

2.2 Avgrensning

Vår oppgave er å lage en applikasjon som beskrevet i neste seksjon (2.3 oppgavebeskrivelse). Vi skal ikke delta i andre funksjoner som markedsføring eller økonomi der Hopla allerede har gjort research for behovet til applikasjonen.
Appen skal i første omgang kun fungere for iOS- og Android-mobilsystemer med et enkelt webgrensesnitt, men skal kunne utvides ved senere tidspunkt om ønskelig.
Det skal ikke være med flere funksjonaliteter rundt hestehold enn rideturer, men heller bygge en skalerbar kodebase slik at appen kan utvides ved senere tidspunkt.

2.3 Oppgavebeskrivelse

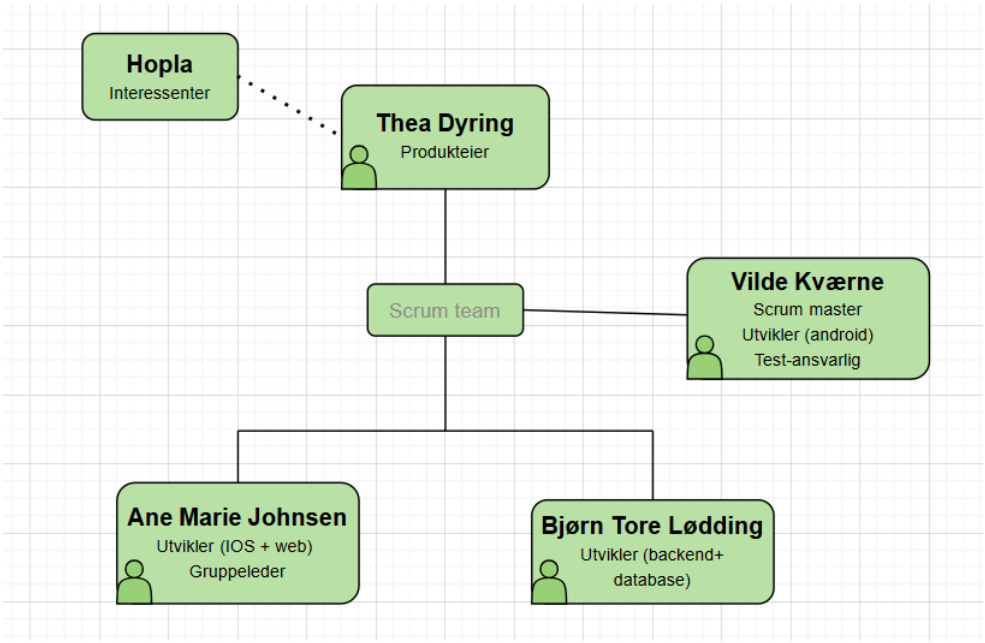
Vi skal levere en fungerende mobilapplikasjon for ryttere, samt et webgrensesnitt som lar oppdragsgiver administrere portalen samt generere statistikk. Appen skal i utgangspunktet kjøres fra Render, men skal være bygd opp sånn at det er lett for Hopla å legge den ut på App Store og Google Play om ønskelig ved prosjektets slutt, uten krav til teknisk kunnskap. Vi har i samarbeid med oppdragsgiver utarbeidet følgende kravspesifikasjon:

- Kart med søkefunksjon og filter.
- Loggføring av turer.
- Beskrivelse av løyper.

- Chat-funksjon.
- Kommentarfelt under løyper for å få nyeste oppdateringer om løypen.
- Kunne starte en tur og få veibeskrivelse og oppdateringer til en løype.
- Filter som: lengde, tid, vanskelighetsgrad, årstid, vogn, område, føre, tempo, parkering, popularitet og om du må krysse bro, elv, eller trafikkert vei.
- Brukerprofil.

Kapittel 3: Prosjektorganisering

3.1 Organisasjonskart



Figur 3.1: Organisasjonskart over prosjektgruppen.

3.2 Ansvarsforhold og roller

Vårt SCRUM-team er organisert i de vanligste rollene og ansvarsområdene:

- Produkteier (Thea Dyring)
  - Representerer firmaet Hopla og er bindeleddet mellom firmaets interesser og SCRUM-teamet.
  - Står som vår hovedkontakt fra firmaet, selv om resten av teamet hennes også er velkommen til alle møter og diskusjoner om applikasjonen.
- SCRUM master (Vilde Kværne)
  - Sørge for at utviklingen går i forventet tempo, ellers iverksette nødvendige tiltak
  - Kalle inn til og lede møter inkludert: sprint planning, review, retropective og daily SCRUM
  - Sørge for at alle parter involvert forstår sin rolle og hva som er forventet av alle parter

- Utviklere (Ane Marie Johnsen og Bjørn Tore Lødding)
  - Utvikle applikasjonen etter fastsatte rutiner, samt følge kodestandarder til brukte kodespråk og verktøy

3.3 Rutiner og regler i gruppen

Gruppereglene ligger i sin helhet som vedlegg A. Noen viktige punkter er:

- Møte opp til alle møter punktlig, og gi beskjed om noe skulle oppstå og man ikke kommer eller er forsinket.
- Alle timer skal loggføres minst 1 gang i uken, og alle gruppens medlemmer er forventet å jobbe minimum 30 timer i uken.
- Referat skal skrives fra alle statusmøter, samt møter med oppdragsgiver og veileder.
- Ved irritasjonsmomenter i gruppen, skal disse tas opp med én gang og diskuteres på en saklig måte.

Kapittel 4: Planlegging, Oppfølging og Rapportering

4.1 Hovedinndeling av prosjektet

Prosjektet består av å lage en mobilapplikasjon for iOS og Android, et webgrensesnitt for administrator, backend og database. Vi skal dele opp ansvarsområder ut fra gruppemedlemmenes interesser og erfaringer.

SCRUM
Vi har valgt SCRUM som vår systemutviklingsmodell for dette prosjektet. Gruppen har god erfaring med denne smidige modellen til mindre prosjekter vi jobbet med, og har funnet ut at SCRUM er en passende løsning til vårt bruk. Med SCRUM kan vi dele opp prosjektet i ulike roller og oppgaver, og vurderte derfor SCRUM til å være den beste modellen da vi alle kan arbeide med vårt eget samtidig som vi får innsikt i de andres arbeid gjennom sprint meetings og reviews.

Andre smidige utviklingsmodeller slik som Kanban og eXtreme Programming (XP) ble også vurdert, men vi valgte vekk XP fordi vi er 3 i gruppen, og parprogrammering er en essensiell del av modellen. Med Kanban bruker man ikke sprinter, og fordi vi ønsker å planlegge når vi skal fullføre hvilke oppgaver, gikk vi heller for SCRUM [15].

Hver sprint kommer til å vare i omtrent 2 uker. Vi har allerede erfaring med å bruke ukentlige sprinter i et lignende prosjekt, og dette ble for ofte. Da fikk vi ikke tid til å fullføre de større oppgavene på kun én uke, og måtte derfor forlenge sprintene til hver andre uke - som fungerte mye bedre. Derfor velger vi å videreføre denne erfaringen til dette prosjektet for å kunne utnytte SCRUM enda bedre.

Den første tirsdagen i hver sprint skal vi ha et sprint planning-møte der vi diskuterer hva vi skal gjøre i neste kommende sprint. Etter hver sprint har vi et sprint review-møte for å se hva de andre i gruppen har fått gjort, om noe ikke ble ferdig og hva som kan forbedres. Daily SCRUM blir gjennomført de dagene vi møtes fysisk. Oppdragsgiver skal også få innsikt i hva vi har gjort de siste 2 ukene, slik at de holdes oppdatert i prosessen, samtidig som de kan komme med egne innspill og få innsikt i endringer som blir gjort og funksjonaliteter som blir lagt til.

4.2 Plan for statusmøter og beslutningspunkter i perioden

Annenhver torsdag fra 18.00 - 19.00 har vi digitale møter med oppdragsgiver, som ved behov eller ønske fra en/begge parter kan gjennomføres som fysisk møte enten i Gjøvik

eller i Oslo.

Møte med veileder skjer hver tirsdag fra 11.00 - 12.00 etter behov for veiledning. Dette gjelder for starten av prosjektet, mens vi utover prosjektet kommer til å innføre møter hver annen uke.

Sprint planning meeting bir holdt første tirsdag i sprinten som figur 6.1 viser senere i dokumentet. Hver sprint er lagt opp til å være omtrent 2 uker, med noen sprinter noe lengre for å tilpasse frister i prosjektet. Sprint review blir dermed siste torsdag i sprinten. Under hver sprint blir daily meeting holdt på de resterende fysiske møtene mellom gruppemedlemmene.

Kapittel 5: Organisering og Kvalitetssikring

5.1 Dokumentasjon, lagring og kildekode

For å sikre at alle gruppemedlemmer bruker 30 timer på prosjektet hver uke, skal alle dokumentere sin timebruk i Excel minst én gang i uken. Kilder man bruker underveis, skal legges inn i rapporten etter at man har brukt dem. Møtereferater skal bli gjort tilgjengelig for alle i gruppen. Vi bruker Overleafs egen History-funksjon for å sikre at vi alltid har en backup av rapporten tilgjengelig om vi skulle gjøre uønskede endringer. I tillegg skal vi laste ned en backup til GitLab hver torsdag. Ansvaret for å ta notater fra møter ligger på alle i gruppen, da det viktigste er at dette blir gjort.

Vi skal gjennomføre brukertester av applikasjonen etter at vi har fått implementert grunnfunksjonaliteten appen skal inneholde. UI og UX er en viktig del av prosjektet, og vi skal derfor under møtene med oppdragsgiver vise oppsett og design så langt, og be om tilbakemeldinger for å sørge for at applikasjonen samsvarer med Hoplas visjon. Vi skal også gjennomføre tester av koden for å oppdage feil og bugs som kan føre til en dårligere brukeropplevelse.

5.2 Faste utviklingsrutiner

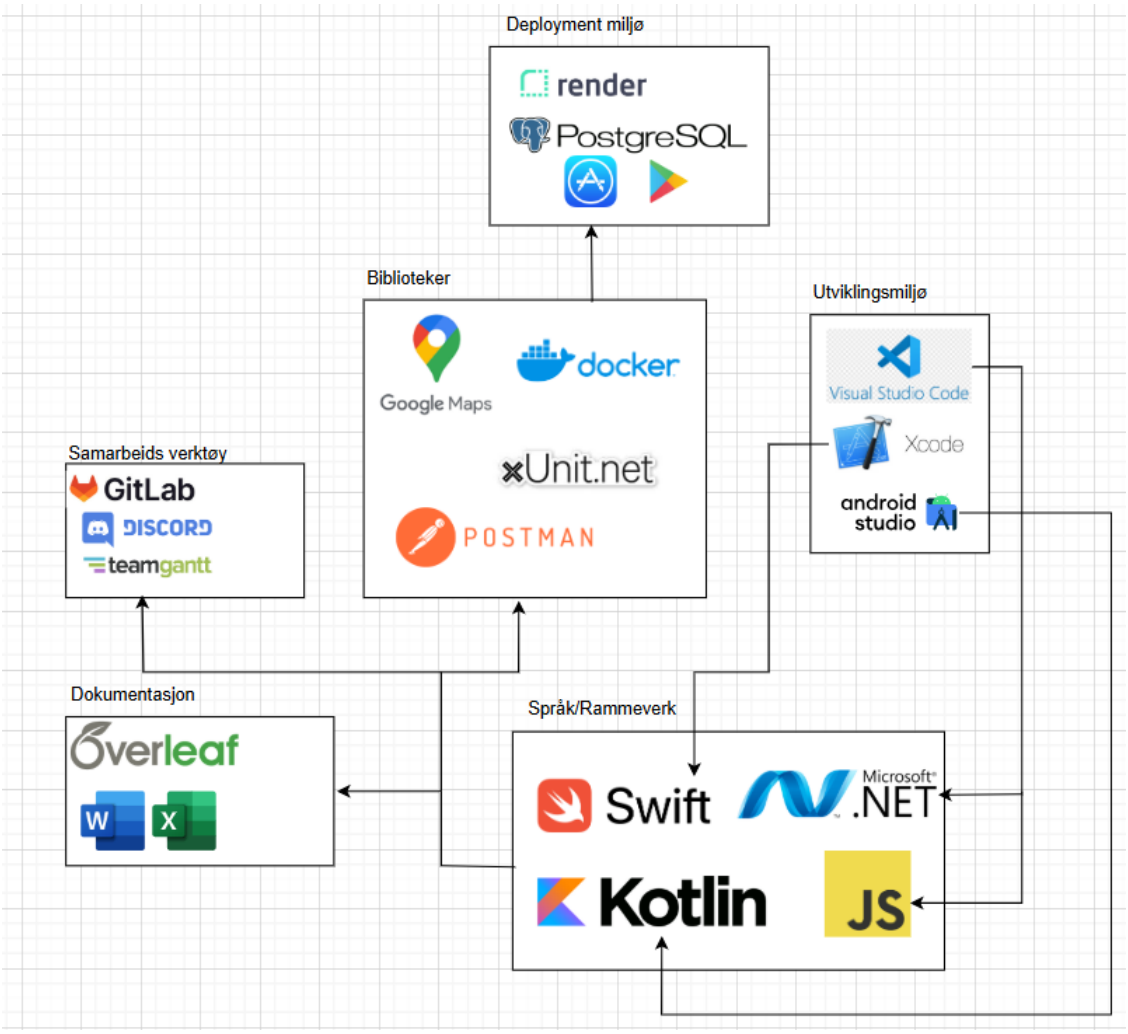
- All kode skal kommenteres før en commiter den til GitLab.
- Commits skal skje jevnlig etter velfungerende endringer / etter hver funksjonsendring.
- Issues skal for hver sprint fordeles ut over gruppens medlemmer.
- Alle skal jobbe i forskjellige branch-er etter hvilken funksjon man jobber med før man merger til main. Det som pushes til main skal alltid være velfungerende før merging.
- Ferdige issues skal legges under "Review" i Gitlab, og gås gjennom av minst én av de andre medlemmene i gruppen før man legger dem under "Closed".
- Når integrasjon- og unit- testing har blitt lagt til, skal man alltid sørge for at alle testene kjøres før man merger til main.
- Ved problemer under utviklingen skal disse tas opp på neste fysiske møte og/eller diskuteres på Discord for å slippe å bruke unødvendig tid på feilsøking alene.

5.3 Verktøy

Tabell 5.1 viser en oversikt over de viktigste verktøyene brukt under utviklingen av programvaren med en kort forklaring på hva de brukes til, mens 5.1 er en grundigere oversikt over alle verktøyene som brukes.

Navn	Type	Bruksområde
OverLeaf	Redigeringsverktøy og kompilator for LaTeX-dokumenter	Rapport
draw.io	Planlegge og tegne diagrammer	Rapport
GitLab	Sammarbeid om kode, scrumboard og prosjektoversikt	Prosjektstyring
Visual Studio Code	IDE for backend-utvikling	Backend
Xcode	IDE for IOS-utvikling	Frontend
Android studio	IDE for Android-utvikling	Frontend
TeamGantt	Online Gantt-verktøy	Prosjektstyring
Render	Lansere applikasjonen	Prosjektstyring
PostgreSQL	Lagring av informasjon om brukere og turer	Database
Postman	Testing av backend	Testing
Docker	Software container platform	Porsjektstyring

Tabell 5.1: Overskrift over verktøy med beskrivelse



Figur 5.1: Visuell oversikt over verktøy som brukes.

5.4 Risikoanalyse

I løpet av prosjektets tidsramme kan det oppstå flere problemer og uventede hendelser. Det er viktig å være klar over dette fra starten av, og å ha tenkt ut mulige løsninger for å sikre at vi ikke kaster bort dyrebar tid og ressurser.
Risikoanalysen under i tabell 5.2 viser hvor sannsynlig det er at en hendelse oppstår, og hvor kritisk hendelsen er for prosjektet.

Nummer	Hendelse	Sannsynlighet	Konsekvens	Tiltak
1	Én eller flere i gruppen blir syke, og kan derfor ikke jobbe med prosjektet i en periode	Sannsynlig	Kritisk	Ja
2	Appen blir ikke ferdig	Lite sannsynlig	Kritisk	Ja
3	Ett eller flere verktøy legges ned eller slutter å fungere	Lite sannsynlig	Kritisk	Ja
4	Ny pandemi	Lite sannsynlig	Ikke kritisk	Nei
5	Oppdragsgiver ønsker et helt nytt konsept langt inn i prosjektet	Lite sannsynlig	Ikke kritisk	Ja
6	Internett og strøm er nede	Sannsynlig	Kritisk	Nei
7	Kode og dokumentasjon går tapt	Lite sannsynlig	Kritisk	Ja
8	Appen finnes allerede	Sannsynlig	Ikke kritisk	Nei
9	Gruppekonflikt	Usannsynlig	Ikke kritisk	Ja

Tabell 5.2: Tabell for risikoanalysen

5.5 Plan for håndtering av risiko

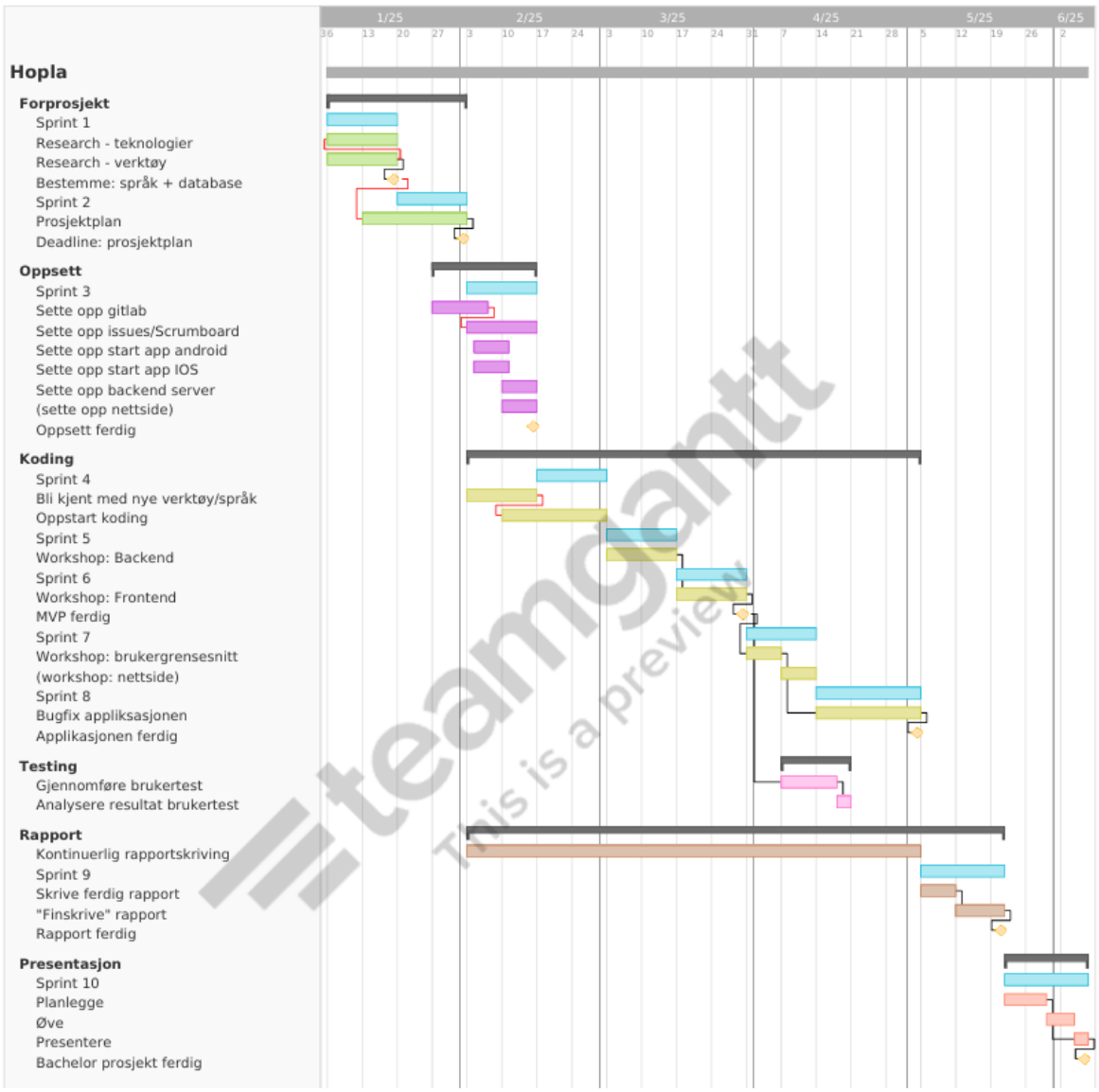
Vi har valgt å gjennomføre tiltak for 7 av de 9 hendelsene som kan oppstå under prosjektet. Hendelsene som ikke krever noen tiltak, er situasjoner som er utenfor vår kontroll eller som ikke spiller noen rolle for utviklingen av appen.
Om en ny pandemi skulle inntreffe, kan vi fremdeles jobbe som vanlig digitalt. Dersom Internett og strøm skulle være nede på grunn av krig eller dårlig vedlikehold, er ikke dette noe vi kan gjøre noe med. Om en tilsvarende app allerede finnes, setter ikke dette en stopper for utviklingen av vår app.De tiltakene vi har satt er vist i tabell 5.3 under.

Nummer	Tiltak
1	Ved å holde et ukentlig møte der vi oppdaterer hverandre om hva vi har gjort, kan de andre i gruppen lett ta over arbeidsoppgavene til det syke gruppemedlemmet. Om to av oss er syke må det siste medlemmet gjøre så godt hen kan, og om mulig legge inn en større arbeidsmengde i den perioden.
2	For å sørge for at appen blir ferdig ved prosjektets slutt, må vi sette klare grenser for hva vi velger å inkludere av funksjonalitet og hva vi må velge vekk. Dette diskuterer vi i startfasen av prosjektet slik at vi tidlig vet hva vi skal fokusere på og har tid til å implementere.
3	Dersom Android Studio, Xcode, Discord eller et av de andre verktøyene vi bruker skulle slutte å fungere, må vi finne noen backup-alternativer. Discord kan byttes ut med Facebooks Messenger-app, Xcode med Visual Studio Code osv. Heldigvis finnes det mange muligheter, og om et av programmene skulle slutte å fungere, er det viktigste å ikke sitte å vente på at problemet fikser seg selv men å se etter en annen løsning.
5	Om oppdragsgiver finner ut langt inni prosjektet at de ønsker noe helt annet enn det vi diskuterte i starten, eller at de vil legge til mye mer funksjonalitet enn tidligere tenkt, er ikke dette vi skal ta noe hensyn til. Prioriteten vår blir å fullføre den originale planen. Selvsagt skal vi høre med oppdragsgiver underveis for å få feedback og høre om forbedringer, men vi har ikke tid til å starte på nytt eller legge til mer funksjonalitet enn det vi planla fra starten av.
7	For å unngå tap av kildekode og dokumentasjon, må alle i gruppen være flinke på å publisere kode til Gitlab, og dette bør gjøres så ofte som mulig etter små og velfungerende endringer. En backup av rapporten tas for hver dag man går inn og endrer eller legger til noe i Overleaf av alle medlemmer automatisk.
9	Oppstår det en konflikt innad i gruppen, skal vi prøve å løse denne ved å kalle inn til et møte. Om vi ikke klarer å løse konflikten ved å snakke sammen, er det opp til gruppeleder å ta en avgjørelse. Hjelper ikke dette, må vi avtale et møte med veileder.

Tabell 5.3: Satte tiltak for risikoanalysen

Kapittel 6: Plan for Gjennomføring

6.1 Gantt-skjema



Figur 6.1: Gantt-skjema for prosjektet.

i	Navn	Start dato	Slutt dato	Forgjenger
1	PROG2900 - Bacheloroppgave	06/01/2025	31/01/2025	
1.1	Forprosjekt	06/01/2025	20/05/2025	
1.1.1	Sprint 1	06/01/2025	17/01/2025	
1.1.2	Research: teknologier	06/01/2025	17/01/2025	
1.1.3	Research: verktøy	06/01/2025	17/01/2025	1.1.2
1.1.4	Bestemme: språk+database	17/01/2025	17/01/2025	1.1.3
1.1.5	Sprint 2	20/01/2025	31/01/2025	
1.1.6	Prosjektplan	13/01/2025	31/01/2025	1.1.4
1.1.7	Deadline: prosjektplan	31/01/2025	31/01/2025	1.1.6
1.2	Oppsett	27/01/2025	14/02/2025	
1.2.1	Sprint 3	03/02/2025	14/02/2025	
1.2.2	Sette opp Gitlab	27/01/2025	05/02/2025	
1.2.3	Sette opp issues/Scrumboard	03/02/2025	14/02/2025	1.2.2
1.2.4	Sette opp start app: Android	04/02/2025	10/02/2025	
1.2.5	Sette opp start app: IOS	04/02/2025	10/02/2025	
1.2.6	Sette opp backend-server	10/02/2025	14/02/2025	
1.2.7	Sette opp nettside	10/02/2025	14/02/2025	
1.2.8	Oppsett ferdig	14/02/2025	14/02/2025	
1.3	Koding	03/02/2025	02/05/2025	
1.3.1	Sprint 4	17/02/2025	28/02/2025	
1.3.2	Bli kjent med nye verktøy/språk	03/02/2025	14/02/2025	
1.3.3	Oppstart koding	10/02/2025	28/02/2025	1.3.2
1.3.4	Sprint 5	03/03/2025	14/03/2025	
1.3.5	Workshop: backend	03/03/2025	14/03/2025	
1.3.6	Sprint 6	17/03/2025	28/03/2025	
1.3.7	Workshop: Frontend	17/03/2025	28/03/2025	1.3.5
1.3.8	MVP ferdig	28/03/2025	28/03/2025	1.3.7
1.3.9	Sprint 7	31/03/2025	11/04/2025	
1.3.10	Workshop: Brukergrensesnitt	31/03/2025	04/04/2025	1.3.8
1.3.11	Workshop: nettside	07/04/2025	11/04/2025	
1.3.12	Sprint 8	14/04/2025	02/05/2025	
1.3.13	Bugfix applikasjon	14/04/2025	02/05/2025	1.3.10
1.3.14	Applikasjon ferdig	02/05/2025	02/05/2025	1.3.13
1.4	Testing	07/04/2025	18/04/2025	
1.4.1	Gjennomføre brukertest	07/04/2025	16/04/2025	1.3.8
1.4.2	Analysere resultat brukertest	17/04/2025	18/04/2025	1.4.1
1.5	Rapport	03/02/2025	20/05/2025	
1.5.1	Kontinuerlig rapportskriving	03/02/2025	02/05/2025	
1.5.2	Sprint 9	05/05/2025	20/05/2025	
1.5.3	Skrive ferdig rapport	05/05/2025	09/05/2025	
1.5.4	Finskriverapport	12/05/2025	20/05/2025	1.5.3
1.5.5	Rapport ferdig	20/05/2025	20/05/2025	1.5.4
1.6	Presentasjon	21/05/2025	05/06/2025	
1.6.1	Sprint 10	21/05/2025	05/06/2025	
1.6.2	Planlegge	21/05/2025	28/05/2025	
1.6.3	Øve	29/05/2025	03/06/2025	
1.6.4	Presentere	04/06/2025	05/06/2025	1.6.2
1.6.5	Bachelor-prosjekt ferdig	05/06/2025	05/06/2025	1

6.2 Milepæler og beslutningspunkter

Vi har satt flere milepæler gjennom prosjektet som definerer hvor langt utviklingen skal ha kommet ved satte tidspunkt. Disse inkluderer:

- **Milepæl 1, 17 januar:** Bestemt kodespråk og database for prosjektet
- **Milepæl 2, 31. januar:** Skrevet ferdig prosjektplan, samt fått igang utviklingsmiljøet
- **Milepæl 3, 3. februar:** Oppstart kodingen
- **Milepæl 4, 28 mars:** Alle grunnkomponenter av applikasjonen skal være fungerende inkludert:
  - fungerende kart med posisjon til brukeren
  - ferdig oppsatt database
- **Milepæl 8, 18. april:** Brukertester ferdig, samt analysert resultatene fra testene.
- **Milepæl 9, 2. mai:** Koden ferdigskrevet og testet
- **Milepæl 10, 20. mai:** Innlevering av hele prosjektet
- **Milepæl 11, 4/5. juni:** Presentasjon av prosjektet

Referanser

[1] Hopla. *Hopla - Kart over rideturer*. Accessed: 2025-01-16. 2025. URL: https://www.hopla.no/.

[2] H. Ulfeng. *Ønsker mer kunnskap om hestenæringa i Norge og Sverige*. Mar. 2023. URL: https://www.nibio.no/nyheter/onsker-mer-kunnskap-om-hestenaeringa-i-norge-og-sverige.

[3] StatCounter Global Stats. *Mobile Operating System Market Share Worldwide*. Accessed: 2025-01-16. Jan. 2025. URL: https://gs.statcounter.com/os-market-share/mobile/worldwide.

[4] A. Avanderlee. "*Minimum iOS Version: How to Decide Which Versions to Support*". I: *Avanderlee* (jan. 2023). URL: https://www.avanderlee.com/workflow/minimum-ios-version/.

[5] Stack Overflow Community. "*What Version of Android Should I Develop For?*" I: *Stack Overflow* (sep. 2010). URL: https://stackoverflow.com/questions/3779572/what-version-of-android-should-i-develop-for.

[6] Tutorials Freak. *HTML History*. Accessed: January 16, 2025. 2025. URL: https://www.tutorialsfreak.com/html-tutorial/html-history.

[7] Movadex. *Native vs Hybrid vs Cross-Platform vs PWA: A Comprehensive Comparison*. Mai 2023. URL: https://movadex.com/blog/article/native-vs-hybrid-vs-cross-platform-vs-pwa-a-comprehensive-comparison.

[8] M. Nenseth, S. Flovik og Y. Salvesen. "*Progressive Web Apps (PWA) – hva er det, og hvorfor trenger du det?*" I: *Aktuelt, Brukeropplevelser og design, Teknologi og digitalisering* (jun. 2020). URL: https://egde.no/aktuelt/progressive-web-apps/.

[9] Integrate.io. *Which Database Should You Use?* Jan. 2025. URL: https://www.integrate.io/blog/which-database/.

[10] Apple Inc. *Swift - Apple Developer*. Accessed: 2025-01-23. 2025. URL: https://developer.apple.com/swift/.

[11] Google. *Kotlin for Android Developers*. Accessed: 2025-01-23. 2025. URL: https://developer.android.com/kotlin/.

[12] Mukesh Ram. *Exploring the Role of HTML, CSS, and JavaScript in Frontend Development*. Accessed: 2025-01-23. Jan. 2023. URL: https://medium.com/@mukesh.ram/exploring-the-role-of-html-css-and-javascript-in-frontend-development-9f3b9123302b.

[13] Microsoft. *Why .NET?* Accessed: 2025-01-23. Sep. 2023. URL: https://devblogs.microsoft.com/dotnet/why-dotnet/.

[14] GeeksforGeeks. *Difference Between SQL and NoSQL*. Accessed: 2025-01-23. 2023. URL: https://www.geeksforgeeks.org/difference-between-sql-and-nosql/.

[15] A. Rolstadås. "*kanban (produksjonsteknikk)*". I: *Store norske leksikon* (okt. 2024). URL: https://snl.no/kanban_-_produksjonsteknikk.

A Grupperegler

Rutiner og regler

- Gruppeleder er Ane Marie.
- Ukentlige møter med veileder i starten av prosjektet, som reduseres til hver andre uke ved behov.
- Referent roteres mellom alle gruppemedlemmene i følgende rekkefølge:

Ane Marie	Vilde	Bjørn Tore

- Alle gruppemedlemmer forplikter seg til 30 timers arbeid hver uke som et minimum, mer enn 30 timer er ikke forventet.
- Timene man har brukt skal loggføres i Excel minimum 1 gang i uken. Link til denne finnes på GitLab.
- Hvis man av god grunn ikke kan møte opp til oppsatte møter, gi beskjed så fort som mulig, minimum 24 timer i forveien med unntagelse av ved sykdom.
- Man er forventet å møte opp til fysiske møter hver uke, tirsdag (09.00-14.00) og torsdag (09.00-13.00 eller 15.00-18.00 i uker med oppdraggiver-møte). Hver torsdag avtaler gruppens medlemmer om disse dagene passer for alle neste uke eller må endres, samt om man har behov for flere/færre møter den kommende uken.
- Hver tirsdag og torsdag i starten av møtet er alle forventet å gi en kort oppsummering av hva de har gjort siden forrige møte som en del av daily SCRUM.
- Ved delte meninger skal man til beste evne prøve å inngå kompromiss, eventuelt må gruppeleder gå inn å ta et valg som gruppeleder mener er best for både gruppen og prosjektets interesser.
- Det er ikke tillatt å spise mat som har vond lukt under fysiske møter. Noen eksempler på slik mat er leverpostei og makrell i tomat. Ved mer alvorlige brudd hvor man har pålegg som består av Haugpølse eller annet som inneholder hestekjøtt, vil det føre til umiddelbar utkastelse fra gruppen.
- Hovedkommunikasjonen skal foregå på Discord mellom de fysiske møtene.
- Ane Marie er satt opp som kommunikasjonsansvarlig med veileder og oppdragsgiver, og er ansvarlig for at alle parter holdes oppdatert om gruppens framgang samt at ønsker fra oppdragsgiver formidles videre.

Rutiner ved regelbrudd

- Ved problemer i gruppen forventes det at dette først tas opp innad i gruppen ved en samtale som bringer frem problemet/problemene på en ryddig måte. Om dette ikke løser problemet/problemene, vil dette tas videre til veileder, deretter emneansvarlig om nødvendig.
- Det kan bli gitt ut straff i form av å kjøpe godteri/snacks til de andre medlemmene, eller man kan velge å isbade hvis personen:
  - har kommet for sent 3 ganger.

- ikke har brukt 30 timer på prosjektet gjentatte ganger.
- ikke gir beskjed om at hen blir for sen til møtet 3 ganger.
- ikke gir beskjed om at hen ikke kommer.

Dette blir registrert i Excel.