Bacheloroppgave

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

Mahmoud Ibrahim

Eirik Plahte

Christian Riksvold

Utvikling av en mobilapplikasjon for loggføring på båt med en sømløs brukeropplevelse uavhengig av nettverksforbindelse

Bacheloroppgave i Bachelor i ingeniørfag, data

Veileder: Donn Morrison

Trondheim

Mai 2021

NTNU
Kunnskap for en bedre verden

Forord

Dette er hovedrapporten til gruppe 50 i emnet TDAT3001 Bacheloroppgave Dataingeniør. Prosjektet ble utført av 3. årsstudenter på NTNUs dataingeniørstudium i vårsemesteret 2021 i samarbeid med Kantega AS, og i regi av Fakultet for informasjonsteknologi og elektroteknikk.

Å utvikle en applikasjon for Android, som har en veldig stor brukerskare, hørtes spennende og lærerikt ut, og er noe som er nyttig å ha med videre i arbeidslivet. Alle teammedlemmene hadde noe erfaring med apputvikling fra før, og var interesserte i å lære mer. Noen av teammedlemmene har seilt en god del og syntes det var spennende og gøy å kunne utvikle en app som kan brukes for noe teammedlemmene er interessert i.

Utviklingen av produktet har vært veldig krevende. Det har vært mange sene kvelder og lange timer med problemløsning, men vi har alltid kommet til en løsning til slutt. Vi har erfart at det er vanskelig å planlegge et så stort prosjekt og samtidig holde oss til planen. Store endringer kan skje underveis som selvfølgelig er naturlig ved smidig utvikling, men har allikevel satt litt kjepper i hjulene våre. Det har uansett ikke stoppet oss fra å utvikle et produkt vi er fornøyde med og kan være stolte over. Dette har vært en erfaring som har gitt oss et godt grunnlag for arbeidsmarkedet og gitt oss mulighet til å utforske flere relevante teknologier.

Teammedlemmene vil takke produkteier Håvard Wigtil for et hyggelig og profesjonelt samarbeid. Produkteieren har vært hjelpsom og involvert med teamet gjennom hele prosjektet. Vi vil også takke vår veileder Donn Morrison som var involvert gjennom hele prosessen og ga oss nyttige tilbakemeldinger.

Sted og Dato

Trondheim 19.05.2021

Mahmoud Ibrahim	Eirik Plahte	Christian Riksvold
		

Oppgavetekst

Oppgaveteksten slik den ble formulert av produkteier er som følger:

> Loggboka er en viktig komponent i et båthold. Den dekker turbeskrivelser, praktisk informasjon med mer, og kan fungere som "hyttebok". Det finnes i dag ingen gode digitale løsninger som dekker alle disse aspektene på en god måte, og ingen som ivaretar det sosiale aspektet som er til stede i båtlivet. Oppgaven er å utvikle en app som har som kjernefunksjonalitet å logge en båtreise med posisjonsspor, og ta tekst og bilder under veis som knyttes til spesielle hendelser. Innholdet skal kunne deles med medreisende og andre.

Implisitt i dette inngår det at vi har et registreringssystem for brukere, der all data lagres i en database. Dette medfører at det trengs en server som kan kommunisere med databasen. Videre må en bruker kunne registrere en båt, og knytte andre brukere til denne båten. Man skal kunne registrere en tur på en båt, og legge til andre brukere som medreisende på denne turen, slik at de også kan skrive logginnlegg. Det skal også være mulig å legge til medreisende som ikke er brukere, slik at disse personene ved en senere anledning kan registrere seg og få tilgang til de tidligere turene de har blitt lagt til i.

Funksjonene som er beskrevet i sitatet over er implementert i det ferdige produktet. I tillegg til kjernefunksjonaliteten som er beskrevet ovenfor har vi også implementert flere tilleggsfunksjoner. Eksempelvis å legge til havner og å legge til logginnlegg på en tidligere passert posisjon og å dermed få riktig tidspunkt til logginnlegget, og motsatt. Det er også mulig å velge en egendefinert startposisjon og tidspunkt dersom man glemmer å starte sporingen når man starter en tur, og tilsvarende med sluttposisjon og tidspunkt dersom man glemmer å stoppe sporingen når man er ferdig med en tur. Brukere kan i tillegg søke etter logginnlegg de har tilgang til, både basert på søkestrenger som brukeren skriver inn i et felt på forsiden og ved å spesifisere en radius rundt nåværende posisjon på en tur, og dermed finne logginnlegg innenfor denne radien.

Det blir nevnt flere mulige tilleggsfunksjoner i det opprinnelige oppgavetekst-dokumentet (se vedlegg) under punktet “Utfyllende kommentarer til hva oppgaven gjelder”, deriblant “nettside for bedre visning av turer” og “sporing av forbruk (diesel, olje, vann)”. Dette er funksjoner vi ikke har hatt tid til å implementere. I tillegg var det lenge tenkt at brukeren skulle ha mulighet til å arkivere flere båtturer i en overordnet reise, etter ønske fra produkteier. Alle de utvidelsene vi har implementert er avtalt med produkteier etter smidige prinsipper, slik det blir beskrevet i oppgavetekst-dokumentet.

Sammendrag

Under bacheloroppgaven har teamet utviklet Android-applikasjonen “I samme båt”, som har som kjernefunksjonalitet å logge en båtreise med posisjonsspor, og knytte tekst og bilder til spesielle hendelser. Innholdet skal kunne deles med medreisende og andre.

Gjennom applikasjonen vil brukerne kunne starte en tur og se posisjonen i sanntid på et sjøkart. Posisjoner blir lagret i en skybasert server (Google Cloud) eller lokalt når brukeren ikke har dekning med tanke på at applikasjonen er ment til å brukes ute på sjøen hvor telefonnettet ikke alltid er pålitelig. De fleste hovedfunksjonaliteter i applikasjonen skal være tilgjengelige når brukeren ikke har dekning. Data blir lagret lokalt og vist til brukerne før de blir sendt til serveren når mobilen har fått dekning igjen. Gjennom applikasjonen kan brukerne legge til familie og venner på båten og gi dem forskjellige roller og rettigheter. Brukere som er tilknyttet båten vil enkelt kunne bli lagt til på fremtidige turer.

Denne rapporten omhandler utviklingen av dette produktet med vekt på funksjonalitet ved mangelfull dekning. Vi skal gjøre rede for teorien som legger grunnlaget for applikasjonen og beskrive hvilke teknologier og metoder vi har valgt samt hvorfor vi valgte dem. Hoveddelen av rapporten tar for seg resultatene fra utviklingen og diskuterer hvorfor produktet og prosessen endte opp slik de gjorde. Avslutningsvis skal vi forsøke å knytte sammen resultatene opp mot vår valgte problemstilling.

Innholdsfortegnelse

Figur- og tabelliste

Ordliste og forklaringer

Android	Mobil-operativsystem, basert på en modifisert versjon av Linux-kjernen
Backend	Den delen av systemet som ligger nærmest databasen og server der data blir lagret
Java	Objektorientert programmeringsspråk
JWT	JSON Web Token
URI	Uniform Resource Identifier. En sekvens av tegn som identifiserer en spesifikk ressurs
Android SDK	SDK står for Software Development Kit på engelsk. Dette ble brukt til å bygge klientsiden av applikasjonen
GitLab	Plattform hvor utviklere kan lagre kildekoden slik at alle kan ha tilgang til den
JVM	Virtuell maskin som konverterer Java bytecode til maskinspråk, og gir et kjøresystem for å kjøre Java-koden
Cache	Lagring av data på klienten slik at den kan hente det derfra istedenfor fra serveren dersom data ikke har blitt endret, noe som reduserer datatrafikk
DAO-klasse	En klasse som inneholder SQL-script til en entitet som endepunktene benytter til å håndtere data i database
IDE	Dataprogram som konverter kildekode (ren tekst) til maskinspråk
Quarkus	En full-stack, Kubernetes-native, Java Application Framework
DevOps	Et sett som tar sikte på å forkorte systemutviklings-livssyklus og gi kontinuerlig levering med høy programvarekvalitet

JUnit	Et rammeverk for å teste programvarekomponenter
Sprint	En fase i prosjektet med bestemt lengde og mål som er del av scrum-utviklingsmetodikken
Produkt-backlog	En liste over prosjektets funksjoner samt prioritering og estimert tid til enhver funksjon
GPS	Global Positioning System
Aktiv båt	Båten som en tur automatisk blir registrert på når en bruker starter en tur. Brukeren kan bytte aktiv båt dersom hen eier eller er deleier av flere båter
SharedPreferences	En singleton-klasse som lever gjennom programmets livssyklus hvor man kan lagre enkelte objekter med nøkkel
SL4JF	Logging Facade for Java gir et Java logging API ved hjelp av et enkelt fasademønster
Aktivitet	Representerer en enkelt skjerm i Android f.eks. “logg inn”-aktivitet
Git-gren	Et unikt sett av kildekodeendringer med eget navn
Commit	En kommando som brukes til å lage endringer av kildekode i lokal gren til enhver utvikler
Merge-konflikt	En konflikt i Git som oppstår når flere utviklere har forskjellig kildekode-versjon og prøver å sammenslå dem
Issue-board	Et verktøy i Git som brukes til å planlegge, organisere og visualisere arbeidsflyt
API	Application programming interface. Et grensesnitt som definerer interaksjoner mellom flere programvareapplikasjoner

Roller på båt

På båter kan brukere ha tre forskjellige roller: admin, regulær og gjest. I tabellen nedenfor vises rettighetene til hver av disse tre rollene.

Rolle	Gjest	Regulær	Admin
Tilgang på turer en selv er med på	Ja	Ja	Ja
Enkelt å legge til som passasjer fra en liste	Ja	Ja	Ja
Lesetilgang på alle turer og logginnlegg på båten	Nei	Ja	Ja
Kan starte en tur	Nei	Ja	Ja
Skrive logginnlegg på båten uten tilknyttet tur	Nei	Ja	Ja
Administrere brukere tilknyttet båten	Nei	Nei	Ja
Redigere informasjon om båten	Nei	Nei	Ja
Slette båten	Nei	Nei	Ja

Figur (0.1): Roller på båt

Brukeren som registrerer båten blir automatisk administrator. Denne brukeren har muligheten til å delegere administratorrettigheter til andre brukere, samt sletting og redigering av alle logginnlegg, turer og selve båten. Brukere med rollen "Regulær" er typisk personer i samme

familie som admin, eller andre medeiere. Disse har alltid tilgang til alle turene på båten og kan også starte egne turer. Brukere med rollen “Gjest” har kun lese- og skrivetilgang på turene de har blitt lagt til i.

Rettigheter på tur

På en tur har vi ulike rettighetsnivå på brukere som er tilknyttet. Dette bygger på en funksjon fra visjonsdokumentet som forklarer at en må kunne “administrere hvem som kan logge ved en tur”. En bruker kan ha en av tre mulige rettighetsnivå: admin, lese- og skrivetilgang eller bare lesetilgang. De forskjellige rettighetene til nivåene vises i tabellen under.

Rettighetsnivå	Lesetilgang	Lese- og skrivetilgang	Admin
Se båten og ruta båten har tatt	Ja	Ja	Ja
Lese logginnlegg	Ja	Ja	Ja
Skrive logginnlegg	Nei	Ja	Ja
Redigere og slette sine egne logginnlegg	Nei	Ja	Ja
Administrere tilknyttede brukere	Nei	Nei	Ja
Redigere og slette andres logginnlegg	Nei	Nei	Ja
Avslutte turen	Nei	Nei	Ja
Redigere ruta og informasjon om turen	Nei	Nei	Ja

Figur (0.2): Rettighetsnivå på tur

En bruker med lesetilgang kan bare se informasjonen. Logginnlegg og ruta er tilgjengelig, men ikke mulig å endre for denne brukeren. Dette er et nivå en bruker får når den blir delt en tur med. En bruker med lese- og skrivetilgang har samme rettigheter som de tidligere nevnte, i tillegg til å kunne skrive, redigere og slette sine egne logginnlegg. Dette gjelder for passasjerer på turen. Det finnes kun én adminbruker på hver tur, og det er den som startet turen. I tillegg til alle de samme rettighetene de andre nivåene har, har admin også følgende rettigheter: endre all overordnet informasjon om turen, redigere og slette alle logginnlegg selv om hen ikke har skrevet dem, administrere hvem som er knyttet til turen og hvilke rettigheter de har og å avslutte turen. I tillegg er det denne brukeren som sporer posisjonene til turen og sender det til de andre.

Brukere kan bli lagt til i turen enten ved å få turen delt med seg eller ved å bli lagt til som passasjer. Etter ønske fra produkteier skal alle som har rollen admin eller regulær på båten ha tilgang til alle turene. Regulære og adminbrukere blir vist i en liste når man skal redigere turen og man kan enkelt markere hvem av dem som er med på turen. Andre brukere som er med på turen må bli lagt til enten via mail eller navn om de allerede er tilknyttet båten som gjest. Om de ikke er merket som gjest på båten fra før, vil de automatisk bli lagt til som det etter at de blir knyttet som passasjer til en tur. Dette er alt etter ønske fra produkteier.

Kapittel 1: Introduksjon og relevans

Opphavet til oppgaven var at produkteier hadde et behov han ville dekke. Han er medeier av en båt, og hadde bruk for en applikasjon til å logge båtreiser som ivaretar det sosiale aspektet ved båtlivet. Han ville ha en applikasjon som lar flere deleiere og medreisende skrive til samme logg, med mulighet for å dele turer med andre. Det er dette som er applikasjonens enestående salgsargument, som man ikke finner i noen liknende eksisterende applikasjoner.

Problemstillingen vi har valgt er som følger: "Hvordan kan vi designe og implementere en nettverksbasert mobilapplikasjon som opprettholder en god brukeropplevelse selv ved redusert eller mangel på nettverksforbindelse?"

Når man er ute på sjøen er det ofte lite eller ingen nettverksforbindelse, og derfor var det viktig for produkteier at applikasjonen skulle fungere så bra som mulig selv under disse forholdene. Dette innebærer blant annet at alle posisjoner som blir sporet og alle logginnlegg som blir skrevet uten dekning skal lagres på mobilen i mellomtiden med mulighet for visning og redigering, og de skal bli lagret i databasen når man får dekning igjen. Målet er at man får en så sømløs brukeropplevelse som mulig, og man skal helst ikke legge merke til noen store forskjeller mellom hvordan applikasjonen er å bruke med dekning kontra uten. All kjernefunksjonaliteten til applikasjonen skal fungere som om man har dekning hele tiden, med noen få unntak, slik som å registrere en ny bruker. Man kan legge til medreisende, men

de vil ikke motta rettighetene før admin får nettverkstilkobling. Disse to funksjonene kan umulig gjennomføres uten kommunikasjon med serveren.

Det finnes noen lignende applikasjoner på markedet fra før, for eksempel “Blue Boat Log”, “Sailsense” og “Sail Expert”. I motsetning til applikasjonen vår bruker ingen av de nevnte appene et sjøkart, men heller et vanlig kart. I tillegg funker ingen av de nevnte appene uten dekning. Noen applikasjoner ga oss tilbakemelding på at mobilen ikke var tilkoblet til internett, mens andre ikke gjorde det. Vi opplevde til og med at noen av applikasjonene frøs da vi slo av nettet. Da vi prøvde å legge til data i de nevnte appene, f. eks. et logginnlegg, så det først ut som at det ble lagret, men da vi slo på nettet igjen var alle dataene vi la til borte. I “Sail Expert” ble dataene ikke borte, men da vi prøvde å åpne et logginnlegg stod det at en feil har oppstått. I tillegg hadde ikke den sistnevnte en knapp for å lagre data, men vi måtte trykke på tilbakeknappen for å få en bekreftelsespopup som spurte om vi ville lagre data eller ikke. Da vi skulle teste “Sailsense” uten nett, ble vi alltid stående i velkomstskjermen uten å komme videre inn i applikasjonen, og vi fikk heller ikke tilbakemelding på at mobilen var i offline modus.

I neste kapittel vil vi gå løs på teorien bak applikasjonen. Mesteparten handler om teknologien vi bruker og hvordan den fungerer, men vi har også litt teori om arbeidsmetodene vi brukte. I kapittel tre skal vi beskrive hvilke metoder og teknologier vi har valgt å bruke for å svare på problemstillingen, og hvorfor vi valgte dem. Kapittel fire beskriver hvilke resultater vi fikk både med tanke på de målene vi satte oss i starten av prosjektet, om arbeidsmetodene var hensiktsmessige og hvordan hvert aspekt ved produktet endte opp. I kapittel fem skal vi diskutere resultatene og finne ut hvorfor og hvordan ting ble som de ble før vi til slutt skal argumentere for en konklusjon satt opp mot problemstillingen.

Kapittel 2: Teori

2.1 Klient og server

Applikasjonen tar i bruk en klient-server arkitektur. Dette vil si at klienten sender forespørsler til serveren når en bruker vil hente, lagre, redigere eller slette data. Serveren sender en respons tilbake som inneholder en statuskode som indikerer hvorvidt forespørselen ble forstått og godtatt av serveren, noe som er avhengig av formatet og innholdet til forespørselen fra klienten. Dersom klienten ønsket å hente data blir dette også sendt tilbake fra serveren (såfremt forespørselen ble forstått og godtatt).

2.2 HTTP-protokollen

Nettverksprotokollen som tas i bruk i forespørslene og responsene fra henholdsvis klient og server er HTTP (*Hypertext Transfer Protocol*, eller hypertekstoverføringsprotokoll på norsk). En HTTP-forespørsel består av en forespørselslinje, headerlinjer og en kropp med eventuelle data. Forespørselslinjen består av en HTTP-metode, stien til den forespurte ressursen, og hvilken versjon av HTTP som brukes. Headerlinjer inneholder diverse informasjon avhengig av hva slags forespørsel som sendes, men den inneholder alltid domenenavnet til serveren. Dersom klienten vil sende data til serveren blir dette sendt i kroppen til forespørselen.

I avsnittet over blir det nevnt at forespørselslinjen inneholder en HTTP-metode. HTTP-metoder formidler hva klienten vil gjøre med den forespurte ressursen, og de er definert i paragraf 4.3 i RFC 7231 (IETF, 2014). GET-metoden forespør en representasjon av den spesifiserte ressursen. GET-metoden er den vanligste måten informasjon blir hentet av klienter på nettet. POST-metoden forespør at serveren prosesserer innholdet i kroppen til HTTP-forespørselen og lagrer det. PUT-metoden forespør at innholdet i den spesifiserte ressursen blir erstattet eller oppdatert med innholdet i kroppen til HTTP-forespørselen. DELETE-metoden forespør at den spesifiserte ressursen blir slettet. Det finnes flere HTTP-metoder enn disse, men de øvrige nevnte de vi bruker i prosjektet.

HTTP-servere bruker et standardisert system for statuskodene som sendes tilbake til klienten. Dette systemet er definert i paragraf 10 i RFC 2616 (IETF, 1999). Dersom klientens forespørsel blir mottatt, forstått og akseptert av serveren, blir en statuskode på formatet 2xx returnert, der x er et siffer fra 0 til 9. 200 OK er den vanligste statuskoden i denne kategorien. Den blir sendt når klientens forespørsel har lyktes. En annen vanlig statuskode i denne kategorien er 201 Created, som blir sendt når klientens forespørsel har resultert i at en ny ressurs har blitt skapt (f.eks. når en ny bruker er registrert i et brukersystem).

Statuskoder med formatet 3xx blir sendt når brukeren må bli videresendt for å fullføre forespørselen som ble sendt. Et eksempel på en slik statuskode er 301 Moved Permanently, som blir sendt hvis den forespurte ressursen har blitt gitt en ny URI, og det indikerer at alle lenker til denne ressursen burde oppdateres med den nye URI-en.

Statuskoder med formatet 4xx blir sendt når forespørselen ikke kan bli fullført, og skylden er på klientens side. Den vanligste av disse statuskodene er 404 Not Found, som blir sendt når den forespurte ressursen ikke blir funnet av serveren. Dette skyldes ofte feilstaving eller andre brukerfeil. En annen av disse statuskodene er 401 Unauthorized, som blir sendt når brukeren ikke er autorisert til å hente den forespurte ressursen.

Dersom forespørselen ikke lykkes og feilen er på serveren sin side blir en statuskode med formatet 5xx sendt. En vanlig statuskode i denne kategorien er 500 Internal Server Error, som

er en generell feilmelding som blir sendt når noe uforutsett skjer med serverens håndtering av forespørselen som forhindrer den fra å bli fullført. En annen statuskode i denne kategorien er 503 Service Unavailable, som blir sendt når serveren ikke kan håndtere forespørsler grunnet overbelastning, vedlikehold eller andre midlertidige forhold.

2.3 HTTPS

I vanlige HTTP-forespørsler blir dataen sendt i klartekst. Dette gjør at dataen er sårbar overfor hackere, som kan fange opp pakkene som sendes og lese hva som står der. Dette kan få store konsekvenser dersom sensitiv data slik som passord blir fanget opp. Derfor er det i dag vanlig å bruke HTTPS (*Hypertext Transfer Protocol Secure*), som er en sikrere utgave av HTTP. HTTPS bruker enten SSL (*Secure Sockets Layer*) eller dens etterfølger TLS (*Transport Layer Security*) for å kryptere data som sendes mellom klient og server (Bartnes, 2019, avsnitt 2). Kommunikasjonen mellom klienten og serveren i systemet vårt bruker HTTPS.

2.4 REST-arkitektur

REST (*Representational state transfer*) er en arkitektonisk stil for distribuerte systemer som ble definert av informatikeren Roy Fielding i hans avhandling fra 2000. REST benytter seg av HTTP-protokollen og har som mål å øke ytelse, skalerbarhet, portabilitet, pålitelighet og simplisitet i klient-server systemer (Fielding, 2000). Dette oppnås ved hjelp av flere begrensninger man setter på et slikt system, og hvis alle disse begrensningene er oppfylt kan systemet kalles for RESTfullt.

Den første begrensningen som kreves er at systemet bruker en klient-server arkitektur. En annen av disse begrensningene er at kommunikasjon mellom klient og server må være tilstandsløs. Dette vil si at hver forespørsel fra klient til server må inneholde all informasjon som er nødvendig for at serveren skal forstå forespørselen, og må ikke benytte seg av noe som er lagret på serveren for at den skal bli forstått. Sesjonsstatus blir derfor lagret på klienten. Hver forespørsel fra klient til server blir da håndtert som om det var den første forespørselen som ble sendt fra akkurat denne klienten. En fordel med dette er at serveren slipper å lagre tilstandsavhengig data fra klienten. En annen fordel er at et REST-API kan

fordeles til flere ulike servere hvor hver av dem kan håndtere en hvilken som helst forespørsel siden de er uavhengige av sesjonsdata.

Muligheten for å cache respons fra server er en annen begrensning som kjennetegner et RESTfullt system. Dersom en respons fra en server blir merket som "cacheable", kan klienten lagre dataen fra responsen til når ekvivalente forespørsler blir sendt til serveren senere (Fielding, 2000). Dette er med på å øke effektivitet i og med at det kan delvis eller fullstendig eliminere visse interaksjoner mellom klient og server.

Et uniformt grensesnitt er et annet kjennetegn ved REST. Dette oppnås blant annet ved hjelp av å bruke HTTP-protokollen til å beskrive kommunikasjon mellom klient og server, og å bruke URI-er til å identifisere ressurser. Responser fra server skal også inneholde informasjon nok til at klienten vet hvordan den skal prosessere det. Dette bidrar til å gjøre systemet lett forståelig og oversiktlig. Den siste begrensningen som kreves i REST-arkitektur er et lagdelt system. Dette kan i praksis bety at et REST-API kjøres på server A, data blir lagret på server B, og forespørsler blir autentisert på server C. En komponent i systemet kan ikke "se" videre enn den komponenten den interagerer med. Denne innkapslingen av komponenter øker fleksibiliteten til systemet ved å gjøre det lettere å oppdatere eller erstatte en komponent uten å nødvendigvis måtte radikalt endre hele systemet.

Systemet vårt oppfyller fire av de fem øvrige nevnte kravene som trengs for at det kan kalles for RESTfullt. For det første bruker det en klient-server arkitektur. For det andre brukes det tilstandsløs kommunikasjon mellom klient og server. Autentiseringen er token-basert (mer om det i kap. 3.1.7), og token lagres på klienten, slik at hver forespørsel fra klient til server blir håndtert som om det var den første. Det brukes også et uniformt grensesnitt med URI-er som identifiserer hver ressurs, og disse URI-ene er strukturert på en hierarkisk og oversiktlig måte. Systemet er også lagdelt i den forstand at serveren er delt opp i DAO-klasser som håndterer direkte kommunikasjon med databasen og route-klasser som mottar forespørsler fra klient, og sender en respons tilbake sammen med eventuelle data som ble hentet fra DAO-klassene.

2.5 Relasjonsdatabaser

Serveren i prosjektet vårt lagrer og henter data i en relasjonsdatabase. Relasjonsdatabaser lagrer såkalte entiteter som er relatert til hverandre via nøkler. En entitet kan visualiseres som en tabell der radene er kalt tupler og kolonnene er attributter. Eksempelvis kan man ha en entitet kalt "Bruker" der en tuppel inneholder informasjon om én enkelt bruker av systemet, mens en attributt i bruker-entiteten kan være navn, e-postadresse og liknende. En entitet har en identifiserende attributt, som kalles primærnøkkelen. Primærnøkkelen er en unik ID som brukes til å identifisere hver enkelt tuppel. En entitet kan også inneholde en eller flere fremmednøkler. En fremmednøkkel er en attributt som skaper et avhengighetsforhold mellom to entiteter. En fremmednøkkel i en bestemt entitet er vanligvis primærnøkkelen i en annen entitet.

En fordel med relasjonsdatabaser er at de lagrer data på en oversiktlig og strukturert måte. Ved hjelp av spørringer kan en bruker hente fra en hvilken som helst tabell, og ved hjelp av JOIN-funksjonen kan man også hente og kombinere data fra ulike entiteter og hente det i én spørring. En annen fordel er at relasjonsdatabaser sikrer dataenes integritet. Hver attributt i en entitet har en spesifikk datatype, og når nye data skal legges til eller oppdateres må datatypen som sendes inn passe med det som er spesifisert av entiteten. Dermed blir det sikkert at alle tuplene inneholder passende data, og dette øker stabiliteten og påliteligheten til databasen.

2.6 Hashing og salting

Dersom passordene til brukerne var lagret som klartekst i databasen, ville dataen deres vært svært sårbar overfor hackere eller andre personer med onde hensikter som har tilgang til databasen. For å ivareta sikkerheten til brukerne av systemet bruker vi derfor hashing og salting av passord. En hashfunksjon er en funksjon som tar inn en tekststreng som input og returnerer en ny tekststreng med en fast lengde. Den resulterende tekststrengen, kalt hashen, skal være helt ugjenkjennelig fra den opprinnelige tekststrengen, og samme input vil alltid resultere i samme output. Når hashfunksjoner brukes til lagring av passord er det hashen som faktisk blir lagret i databasen. Hver gang en bruker skal logge inn blir passordet hashet. Hvis den resulterende hashen er lik hashen som ligger lagret i databasen i raden til denne spesifikke brukeren, får brukeren logge inn. Dermed slipper man å lagre passordet i klartekst.

En velfungerende hashfunksjon har tre viktige egenskaper. For det første er det en enveis-funksjon; det skal ikke gå an å finne ut av et passord hvis man har hashen (*Password Hashing*, u.å.). For det andre skal det være så nærme som mulig et en-til-en forhold mellom alle mulige tekststrenger og alle mulige hashverdier. Med andre ord: det skal ikke være mulig at to forskjellige passord produserer den samme hashen. Det finnes ikke noen hashfunksjoner som aldri vil produsere slike såkalte kollisjoner, men de bør være så sjeldne som mulig. Den siste viktige egenskapen er at en liten forandring i den opprinnelige tekststrengen burde forandre den resulterende hashen fullstendig.

For å øke sikkerheten ytterligere benytter vi oss av salting av brukernes passord. Det vil si at vi legger til en tilfeldig generert tekststreng, kalt et salt, til passordet før det sendes gjennom hashfunksjonen. Et nytt salt genereres for hvert nye passord, og saltet lagres i databasen sammen med hashen som ble generert etter at passordet og saltet ble satt sammen. Ved senere brukerautentisering blir passordet som brukeren skriver inn slått sammen med saltet som ligger i brukerens rad i databasen. Den resulterende strengen blir hashet, og resultatet blir sammenliknet med hashen som ligger i databasen.

Ved å salte brukernes passord vil to like passord produsere to forskjellige hashverdier, siden saltet som legges til er tilfeldig generert og unikt for hver enkelte bruker. Dette gjør at dataen til brukerne vil bli tryggere fra visse angrep, for eksempel fra angripere i besittelse av regnbuetabeller. En regnbuetabell er en forhåndsberegnet tabell med passord og korresponderende hashverdier. Siden salting av passord medfører unike hashverdier, vil ikke angrep med regnbuetabeller ha noen effekt, med mindre angriperen har en regnbuetabell for hvert mulige unike salt. Dette er imidlertid praktisk talt umulig grunnet den store mengden mulige kombinasjoner av tegn som kan danne et salt.

2.7 Agil utviklingsmetodikk

Agil utvikling (også kalt smidig utvikling) er en paraplybetegnelse på forskjellige utviklingsmetodikker og rammeverk som brukes i programvareutvikling, der det fokuseres på tilpasning i planlegging og fleksibilitet. Agil utvikling ble popularisert av *Manifesto for Agile Software Development* (heretter kalt det agile manifestet) som ble signert og publisert av

sytten programvareutviklere i 2001. Det agile manifestet består av fire følgende punkter (Beck, K. *et al.*, 2001, avsnitt 2):

1. Individer og samspill fremfor prosesser og verktøy
2. Fungerende programvare fremfor omfattende dokumentasjon
3. Samarbeid med kunden fremfor kontraktsforhandlinger
4. Å reagere på endringer fremfor å følge en plan

De fleste agile utviklingsmetodikker er iterative og inkrementelle. Det at de er iterative vil si at de består av prosesser (eller iterasjoner) som gjentas flere ganger i løpet av et prosjekt, der én iterasjon har fast lengde og spesifikke mål som skal oppnås i løpet av denne iterasjonen. Det at de er inkrementelle vil si at produktet blir designet, implementert og testet inkrementelt, altså at det bygges videre på trinnvis. Prinsippene fra det agile manifestet har inspirert en rekke ulike utviklingsmetodikker, deriblant Scrum (mer om det i kapittel 3.2.1), Extreme Programming (XP), Kanban og Lean Startup.

2.8 Posisjonssporing

En av de viktigste teknologiene som brukes i prosjektet vårt er posisjonssporing, nærmere bestemt GPS (*Global positioning system*). GPS er et satellittnavigasjonssystem som ble utviklet av USAs forsvarsdepartement i 1993, og det gir brukere en svært nøyaktig tredimensjonal posisjon. Systemet er basert på 24 satellitter i seks ulike baneplan, og sammen med flere kontrollsentre på jorden sørger de for stabil og tilnærmet lik ytelse nesten uansett hvor på jorden man befinner seg - med unntak av under vann og inne i større bygninger. (Forsell og Kjerstad, 2021, avsnitt 3).

GPS sin virkemåte består i å måle avstanden til minst tre av de øvrige nevnte satellittene samtidig. Siden man kjenner til de nøyaktige posisjonene til satellittene, vil de målte avstandene beskrive nøyaktige posisjoner på jorden. Avstanden som blir kalkulert av satellittene er basert på måling av tiden det tar å overføre et radiosignal fra en satellitt til mottakeren (Forsell og Kjerstad, 2021, avsnitt 5). For at målingene skal bli nøyaktige brukes det ekstremt nøyaktige atomur som befinner seg i satellittene.

Kapittel 3: Valg av teknologi og metode

3.1 Systemutvikling

I dette kapitlet skriver vi om teknologier og metoder som ble benyttet til å bygge applikasjonen samt begrunnelse bak valgene som ble tatt. I tillegg skriver vi om metodikken, arbeidsrutiner og rollefordelingen i løpet av utviklingsprosessen. De fleste metoder og teknologier som har blitt brukt i dette prosjektet ble tatt etter avtale og diskusjon mellom alle teammedlemmer og produkteieren. Formålet med avgjørelsene som ble tatt er å levere et bra produkt til brukerne, tilpasse visjonsdokumentet, optimalisere utviklingen og at teammedlemmene får noen utfordringer som er nyttige å ta med videre i arbeidslivet.

3.1.1 Android

For å utvikle applikasjonen har vi brukt Android-SDK. Grunnen til at vi har valgt å bruke Android-SDK, er at produkteier ønsket å bygge applikasjonen med fokus på Android. Android er et av de mest brukte operativsystemene for mobiler i dag, og android-utvikling er populært blant utviklere. Derfor er det veldig relevant erfaring å ta med seg videre i arbeidslivet. Android SDK er utviklet av Google for Android-plattformen og inneholder Android Studio som IDE, utviklingsverktøy og har eksempler på kildekoder for hvordan man kan bygge en Android-applikasjon. Android har både Java og Kotlin som offisielle språk. Teamet har bestemt å bruke Java som programmeringsspråk både på klientside og serveren.

Grunnet til dette valget er at alle teammedlemmene har god erfaring med dette språket siden vi har lært om de grunnleggende konseptene om programmering i Java.

3.1.2 Google Cloud

Kjernefunksjonaliteten i applikasjonen er posisjonssporing og å kunne tilknytte tekst og bilde til en hendelse. Til å teste dette var vi nødt til sette opp en server som vi kunne koble til våre mobile enheter til, i motsetning til en lokal server som ikke er tilgjengelig utenfor den maskinens system. Det ble diskutert to alternativer mellom teamet og produkteier. Enten en fysisk server eller en skybasert server. En fysisk server krever mye tid og har høye kostnader med tanke på all maskinvare som må settes opp. I tillegg hadde vi måttet koble den mot prosjektet og forandre på kildekoden slik at det passer det nye oppsettet. Vi bestemte oss derfor heller for å gå for en skybasert server i form av Google Cloud. Google Cloud er en virtuell server som støtter MySQL. Da trengte vi ikke å endre på SQL-spørringer siden vi allerede brukte MySQL-serveren som vi fikk fra NTNU. I motsetning til en fysisk server trengte vi ikke noe utstyr og maskinvare for å koble til den. I de siste årene har skybaserte løsninger blitt mer og mer populære blant utviklere, og dette er et pluss å ha med videre i arbeidslivet. I tillegg hadde vi allerede brukt Quarkus på server-siden av prosjektet (se kapittel 3.1.1 i *Systemdokumentasjo*n). Quarkus muliggjør deploying av server til Google Cloud Platform. Google Cloud server ble satt opp og koblet mot prosjektet vårt i samarbeid med produkteier.

3.1.3 MySQL

MySQL er et håndteringssystem for relasjonsdatabaser. Relasjonsdatabaser er definert i kapittel 2.5. Tabellene kan knyttes sammen ved hjelp av nøkler som er en felles kolonne i to eller flere tabeller, eller i en felles hjelpe-tabell som inneholder nøkler om fra tabellene. Dette gjør det lettere for utviklere å håndtere data som ligger databasen enn å lagre all data i bare én fil som i en dokumentdatabase. Teammedlemmene har også god kunnskap om MySQL siden alle har hatt flere fag om dette temaet, og alle får tilgang til en MySQL-server fra NTNU. Derfor har teamet blitt enige om å bruke MySQL.

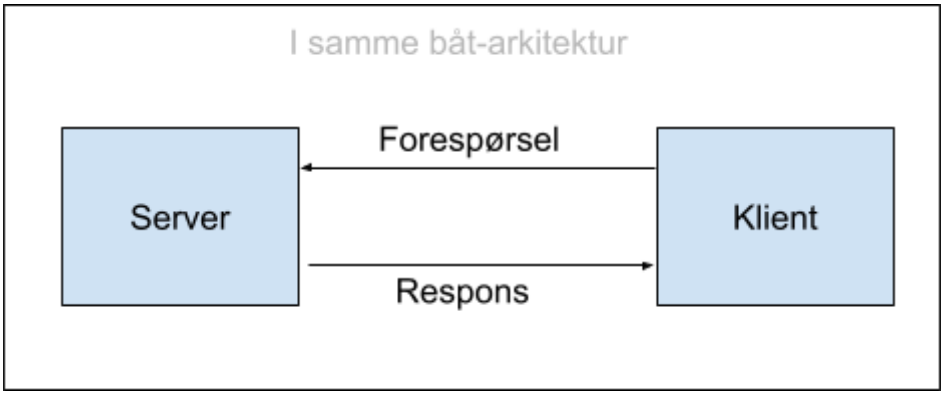
3.1.4 GitLab

Gitlab er en git-basert plattform som implementerer mange verktøy for systemutvikling og håndtering av prosjekter. Gitlab muliggjør at alle teammedlemmene har tilgang til kildekoden. Hvert teammedlem kan fremdeles jobbe med kildekoden lokalt uten å påvirke andre sin kode før endringer blir pushet til git, slik at alle kan oppdatere kildekoden og får den oppdaterte versjonen. Gitlab gjorde det lettere for teamet å se gjennom koden og utviklinghistorikken slik at feilene lett kunne oppdages og rettes. Under utviklingen ønsket vi å ha kontinuerlige tester slik at vi kunne teste endringer av koden i hver commit og oppdage hvilken del av koden som feiler. Teammedlemmene hadde erfaring med innebygd kontinuerlig testing i Gitlab, og alle har en Gitlab-konto som NTNU har skaffet for studentene. Derfor har teamet bestemt å bruke denne plattformen.

3.1.5 Kontinuerlig testing (CI)

Som nevnt i forrige delkapittel har teammedlemmene erfaring med Gitlab-tjenesten kontinuerlig testing, også kalt "CI/CD". Gitlab sin kontinuerlige testing introduserte utviklere til bruk av Docker-bilde. I praksis betyr det at ved bruk av docker-bildet får utviklere en linux-server med all programvare som trengs for å kjøre testene. På den måten kan utviklere sikre at produksjonsmiljøet og testmiljøet ikke er forskjellige. En annen fordel er at teammedlemmene kunne jobbe med funksjoner isolert og har forsikringer om at koden deres sømløst vil integreres med resten av kodebasen, som er en kjerne i DevOps-prosess. Derfor endte vi opp med å bruke denne tjenesten (les mer i kapittel 10.2 i *Systemdokumentasjon*).

3.1.6 Klient-server arkitektur



Figur (3.1): I samme båt - systemarkitektur

Siden teammedlemmene har bygget systemet fra bunnen av, måtte det tas en avgjørelse om hvilken arkitektur som skulle brukes, og måten klienten og serveren skulle kommunisere med hverandre på. Klient-server arkitektur muliggjør at data lagres i serveren som behandler forespørsler fra klienter og sender resultater etter å ha validert brukerautentisering. Arkitekturen er en distribuert modell og er uavhengig av hvilket operativsystem som brukes til selve programvaren, som muliggjør at oppgradering av server eller håndteringen av alle forespørsler skjer uten at klienten får vite om dette. Derfor har teamet bestemt å bruke denne arkitekturen for prosjektet. Ulempen med denne arkitekturen er trafikkbelastning. Hvis et stort antall klienter sender forespørsler til serveren samtidig, kan det føre til problemer. Vi har ikke så store datamengder som sendes i hvert kall, og vi har prøvd å redusere antall operasjoner som kreves i hvert kall til serveren. Dette problemet kan unngås ved å bruke Docker til å automatisk sette flere containere og bilder som kan kjøres i perioder med mye trafikk.

3.1.7 JSON Web Token (JWT)

I starten av prosjektet diskuterte teamet innloggingssystem med produkteieren for å finne en måte for serveren å autorisere brukere. Alternativer teammedlemmene hadde tenkt på var å logge inn via Google, Facebook eller med mail og passord. Produkteieren foretrakk det siste og argumenterte for at det er en del brukere som ikke foretrekker å bruke Google eller Facebook-konto på andre applikasjoner. Teamet var enig med det produkteieren ønsket og måtte dermed selv utvikle autentiseringssystemet. JSON Web Token (JWT) genererer en nøkkel kalt en “token” ved innlogging som skal videre brukes hver gang klienten skal kommunisere med serveren. På den måten slipper brukeren å skrive inn e-postadresse og passord hver gang skal sende en forespørsel. Ved hver forespørsel til serveren blir tokenet validert før det blir sendt ved en oppdatert versjon med ny gyldighetstid. Denne gyldighetstiden kan utviklere sette selv, og etter ønske fra produkteier ble denne satt til ett kvarter.

3.1.8 Personvern og lagring av data

Som nevnt i forrige avsnitt, var det vi som utviklet innloggingssystemet, og derfor var det vårt ansvar hva slags informasjon som ble lagret om brukeren. Dette har vi tatt hensyn til ved å se på GDPR-prinnsipper. Dataminimering innebærer å kun samle inn personopplysninger

som er nødvendige (Personopplysningsloven, 2016), og derfor trenger brukeren bare navn og e-postadresse for å opprette en konto i applikasjonen. Når det gjelder riktighet, så har vi implementert funksjoner for sletting og redigering av brukerinfo, konto, passord og alt annet som brukeren legger til i applikasjonen. Lagringsbegrensning og samtykke har vi tatt hensyn til i løpet av hele utviklingen. F.eks. ved registrering av en ny båt er det kun navnet på båten som obligatorisk å skrive, mens bilde og registreringsnummer er valgfritt. Dette informerer vi brukerne om ved å markere valgfrie felter. Når det gjelder posisjonssporing, så skjer det kun når brukeren har en pågående tur og har gitt samtykke. Det samme gjelder for tilgang til både kamera og galleri. Brukeren kan også legge inn bilder for å representere sin egen profil eller båt, men dette er fullstendig valgfritt.

Serveren tilgjengeliggjør kun data til brukere som er autoriserte og har tilgangsrettigheter. Passordet blir saltet og hashet, og ikke bare lagret som ren tekst i databasen for å forsikre oss mot et sikkerhetsbrudd. Vi bruker også HTTPS som krypterer data ved overføring til server (les mer om hvordan dette funker i kapittel 7.1 i *Systemdokumentasjon*).

3.1.9 Quarkus, H2 og JUnit5

For testing ønsket produkteier å ha integrasjonstesting, som i vårt tilfelle ble å teste responsen fra endepunktene i stedet for enhetstesting. For å få til dette brukte vi Quarkus-biblioteket på serversiden sammen med JUnit5. Quarkus hadde et test-bibliotek inkludert fra da vi satte det opp. Ved hjelp av dette biblioteket og JUnit5 kunne vi dermed teste HTTP-statuskoder i JVM-modus. JUnit5 gir oss “metodemerker” som lar oss styre rekkefølgen og oppsettet på testene. Vi valgte å bruke disse bibliotekene fordi det passet med Quarkus rammeverket vi allerede hadde og produkteier hadde erfaring med det.

Teamet ønsket å automatisere testene mest mulig uten å påvirke de originale dataene som lå i databasen. Til dette brukte vi en H2-database. H2 muliggjorde for oss å kjøre testene i minnet uten å påvirke de originale dataene. I tillegg kan den kjøre en server basert på en fil som kun lever under kjøring av testene. Før hver test kjørte vi filen som inneholdt SQL-scriptet vi hadde fylt med testdata. Måten å håndtere data på i H2 var nesten likt som i MySQL. Derfor ble teamet enige sammen med produkteier om at dette var det beste valget for oss.

3.1.10 Espresso og JUnit4

På klientsiden hadde vi flere alternativer for å teste applikasjonen, men som nevnt i forrige avsnitt, ønsket produkteieren at vi fokuserte på integrasjonstesting, altså å teste sammenhengen mellom forskjellige moduler og komponenter. For å få til dette brukte vi JUnit4 sammen med Espresso som rammeverk. JUnit4 muliggjorde for oss å fortelle koden hvilken rekkefølge testene skal kjøres i og hva som skje før og etter hver test eller alle testene. Dette blir implementert ved hjelp av JUnit4 med testmetoder som kan markeres med blant annet "@BeforeEach", "@Ruler" og "@FixMethodOrder()". Espresso muliggjorde for oss å simulere at brukeren klikker på forskjellige komponenter i applikasjonen og så på hvordan de kommuniserte sammen. I tillegg kunne vi bruke asynkrone metoder i testene, som vil si å vente på at en komponent har blitt initialisert før vi sjekker verdiene i komponeneten. Derfor var de to rammeverkene veldig nyttige ressurser for oss under klienttesting. Ulempen med denne type testing er at den er treg og tok lang tid til både å kjøre og å feilsøke testene.

3.1.11 WireMock

I forrige avsnitt ble det skrevet at vi har implementert integrasjonstesting på klienten. For å få gjort dette trengte vi data fra serveren som Android-komponentene er avhengige av, men vi ville ikke kjøre testene mot den originale serveren vi bruker. Vi ville helst kjøre klienttestene uavhengig av serveren og fortsatt få hentet data som ligger i den. Derfor brukte vi WireMock, som er en simulator for HTTP-baserte API-er. Under en kjøring av applikasjonen hvor vi gikk gjennom alle scenarioer og funksjoner vi ville teste, gjorde WireMock opptak av alle svar og tilbakemeldinger fra serveren. Etter det kunne vi kjøre testene mot WireMock og motta de tilhørende responsene fra forespørslene og feilsøke dem uavhengig av serveren siden alt var lagret i WireMock (les mer om WireMock i kapittel 10.1.2 i *Systemdokumentasjon*).

3.1.12 MapBox og Kartverket

Fra starten av prosjektet ønsket produkteieren at visning av posisjonssporing og turrute skulle være på et sjøkart med fokus på norgeskysten. Mapbox er en kjent plattform blant utviklere som driver med kart og stedstjenester. I tillegg har Kartverket API-er for norgeskysten som kan brukes sammen med Mapbox. Begge plattformer har API-er med åpen kildekode som utviklere kan bruke. Derfor ble det bestemt å bruke denne tjenesten.

Måten sjøkartet blir hentet er via en URL som skal gi dataen på riktig format til kartkomponenten. URL-en vi endte opp med å bruke er: "https://wms.geonorge.no/skwms1/wms.sjokartraster2?&service=wms&version=1.3.0&request=GetMap&bbox={bbox-epsg-3857}&transparent=true&width=512&height=512&layers=all&CRS=EPSG:3857&format=image/png". Oppbyggingen av denne fungerer som et kodespråk for seg selv og er definert forskjellig for hver tjeneste. Ved å skrive inn starten på URL-en sammen med "&request=getInfo" i en nettleser vil man få et langt dokument tilbake med detaljer for hvordan linken skal oppbygges. Her defineres alle attributtene med hvilke muligheter tjenesten kan tilby:

- Layers: Tjenestene har flere lag (layers) som kan vise ulike ting tjenesten har å tilby. I URL-en spesifiseres hvilket lag man vil hente ut fra tittel.
- CRS: Betegner hvilken projeksjon man vil hente. Vi bruker EPSG:3857, som tilsier at vi vil hente merkatorprojeksjon fra tjenesten.
- Service: Betegner hvilken type tjeneste vi vil bruke. WMS (Web Map Service) er den vi bruker og skal hente raster, som er en form for bilder satt sammen i et koordinatsystem.
- Transparent: Tilsier om den delen av verdenskartet som ikke er dekket av tjenesten skal være gjennomsiktig eller ha en bakgrunn.
- Format: Forteller hvilket format vi vil at dataen skal være i.
- BBox: Står for bounding box og betegner hvilken del av kartet vi skal vise. Her bruker vi en egendefinert variabel istedenfor å bruke statiske verdier for å la brukeren ha mulighet til å endre hvilken del av kartet som skal vises.
- Height og width: Definerer størrelsen på visningen.

3.1.13 Justinmind

Det er flere verktøy og teknologier som er tilgjengelige i dag som kan brukes for å visualisere hvordan vi har tenkt at app-struktur, design og funksjonalitet skal se ut. For å få til dette på en effektiv og rask måte har teamet valgt å bruke Justinmind. Justinmind er et verktøy som har hjulpet oss med å lage en prototype. Programmet lot oss enkelt sette opp en struktur med knapper og tekstvisninger som gjorde kommunikasjonen med produkteier angående struktur og design enklere. Komponentene er klikkbare som vil si at prototypen er interaktiv.

3.1.14 Zoom og Discord

Vi hadde et møte med produkteieren hver uke og med både veilederen og produkteieren annenhver uke. På grunn av restriksjoner knyttet til corona-pandemien ble mesteparten av kommunikasjonen med dem utført over nett, og Zoom ble valgt som kommunikasjonskanal. I tillegg brukte vi Discord som kommunikasjonskanal mellom hverandre når det trengtes. Ved bruk av begge kommunikasjonsverktøyene kunne vi lett sette opp møter, og det ga oss mulighet til å dele skjerm som var nyttig for oss i sprint-demoer.

3.2 Utviklingsprosess

3.2.1 Scrum

Teammedlemmene hadde tidligere erfaring med Scrum og likte kjerneprinsippene dens, som er å fordele store prosjekter til flere små delprosjekter og å ha god kommunikasjon mellom teamet og produkteieren. Scrum er en iterativ og smidig utviklingsmetodikk for systemutvikling. Den strukturerer og deler opp utviklingen i flere sykluser som blir kalt sprinter. Sprinter har egen backlog med funksjoner som er plukket ut fra en produkt-backlog som inneholder alle produktets funksjoner. Sprintlengden kan være mellom 1 og 4 uker og hver sprint avslutter på en bestemt dato med en demo av det som ble gjort i løpet av sprinten med produkteier tilstede.

Teamet hadde et møte med produkteieren hver uke og med både veilederen og produkteieren annenhver på slutten av hver sprint (som regel annenhver uke). Ved bruk av scrum-metodikk har teammedlemmene gjort følgende:

- Involvere produkteieren og veilederen i store deler av prosjektet
- Ført god dokumentasjon av kommunikasjon mellom dem og teamet
- Delt opp prosjektet i små deler og sortert på dem etter prioritering gjort i sammarbeid med produkteieren
- Laget burndown-chart for å sammenligne tiden vi estimerte på hver sprint mot den faktiske framgangen vi hadde på funksjonene
- Hadde gjennom prosessen sprintmål som hjalp teamet til å jobbe effektivt og være bestemt på hva som skal gjøres i løpet av utviklingen slik at systemet ble bygd steg for steg

- På slutten av hver sprint hadde teamet en demo av det som ble gjort og fikk tilbakemelding på det fra produkteieren og veilederen
- Har hatt sprint-avslutning og sprint-retrospektiv hvor teamet gikk gjennom det som var bra, det som kunne gått bedre, og hva som kan unngås, gjøres eller endres for å oppnå bedre resultater i neste sprint
- Etter hver sprint hadde vi en kildekode som var kjørbar

3.3 Arbeidsrutine og rollefordeling

3.3.1 Rollefordeling

Siden teamet består av kun tre teammedlemmer, ble teamet enig om å ikke definere spesifikke roller eller gi et ansvarsområde til kun ett teammedlem. Arbeidsoppgavene skulle fordeles likt og på en rettferdig måte slik at alle teammedlemmene fikk kunnskap om de ulike arbeidsområdene. Dette hjalp teamet til å diskutere ulike oppgaver gjennom hele prosessen, og har optimalisert samarbeidet mellom teammedlemmene.

3.3.2 Arbeidsrutine

I starten av planleggingsfasen ble teammedlemmene enige om å ha fire faste arbeidsdager med fast arbeidstid fra kl. 09:00 til 16:00. Arbeidsdagene ble utvidet til fem dager etter at vi ble ferdige med en eksamen vi hadde i et annet fag. Fast arbeidsrutine har hjulpet oss til å ha faste felles pauser som vi brukte til å legge bort prosjektet og finne på noe gøy sammen. Vi har derfor hatt god stemning mellom oss hele tiden, som igjen optimaliserte jobben og gjorde det lettere for teammedlemmene å samarbeide for å løse problemer og oppgaver. Gode arbeidsrutiner har også styrket kommunikasjonen mellom teammedlemmene og redusert misforståelser.

3.3.3 Google Drive

Google Drive er en synkroniseringstjeneste som muliggjør å samarbeide i samme mappestruktur. I disse mappene kan man blant annet opprette tekstbehandlingsfiler der flere brukere kan redigere samtidig. Tjenesten har vært nyttig for teamet og ble brukt mye under

utviklingen. Alle felles filer bortsett fra kildekoden ble skrevet og lagret ved hjelp av denne tjenesten. Vi har strevet for å ha en god og ryddig struktur på mappene.

Kapittel 4: Resultater

4.1 Vitenskapelige resultater

I dette delkapitlet beskriver vi delene av produktet og designet som har direkte med problemstillingen vår å gjøre, i tillegg til resultater fra brukertestene. Resultatene skal kun beskrives, mens diskusjon og vurderingen av innholdet i dette punktet skjer i delkapittel (5.1).

4.1.1 Lokal lagring

For å få appen til å fungere like godt uten dekning har vi implementert lagring av data lokalt på brukerens enhet. Hver gang en forespørsel blir kjørt sjekker vi om mobilen har tilgang til nett, og om den har det utfører vi kallet mot serveren og mottar en respons som vanlig. Hvis forespørselen er av typen GET lagrer vi dataen i en fil før vi utfører det vi ba om dataen for. Ved nye forespørsler til samme URI, vil filen overskrives med den nye responsen vi mottok. Om appen derimot ikke har nettverkstilkobling henter den dataen vi lagret ved forrige kall og bruker den for å gjøre samme nytten. Hvis det er en annen type forespørsel enn GET, altså POST, PUT eller DELETE, blir ikke responsen lagret om vi har nettverkstilkobling. Disse kallene lagres derimot når vi ikke har dekning for å ikke miste data brukeren fyller inn. Alle POST-, PUT- og DELETE-forespørsler blir lagret i dertil egnede filer for så å bli kjørt når klienten igjen oppnår tilkobling mot internett. Alle filer vi lagrer har navn som starter med brukeren sin ID. Dette er for å skille mellom ulike brukere som kanskje bruker appen på samme mobil.

Det er en fin linje mellom å ha på plass så mye funksjonalitet som mulig og å ikke bruke for mye lagringsplass på mobilen. For å forhindre dobbeltlagring i offline modus, legger vi derfor ikke til data fra POST-forespørsler som skal sendes i filer med GET-responser, men vi lagrer det heller bare i en egen fil for POST-forespørsler. Ved visning vil vi heller spleise POST- og GET-filen og vise dataene fra begge to. Dette gjør all endring av data mye mer komplisert da vi både må sjekke gjennom de lagrede responsene fra GET-forespørsler og i tillegg filene med forespørsler som skal postes. I tillegg har vi et system for å legge til nye entiteter offline hvor vi gir entitetene en midlertidig negativ ID. Om vi skal kunne endre på nye entiteter som blir lagt til når vi er i offline modus må vi kunne finne igjen POST-forespørselen i en fil. Derfor gir vi dem en midlertidig ID som skal sørge for at klienten endrer riktig data. Filene som holder data med PUT-forespørsler må da også ha denne midlertidige ID-en som oppdateres så fort POST-forespørselen har blitt sendt og håndtert på serveren. Responsen fra serveren inneholder en reell ID som må settes i filer som er avhengige av dataen som nettopp ble sendt. De fleste PUT- og DELETE-forespørslene som skal endre på data som ikke har blitt postet enda, kan gjøre endringer direkte på POST-filene. Det vil si at om for eksempel en tur blir opprettet ved et uhell og slettet like etter, vil vi ikke sende noe til serveren, men vi vil slette forespørselen før den blir sendt.

4.1.2 Posisjonssporing

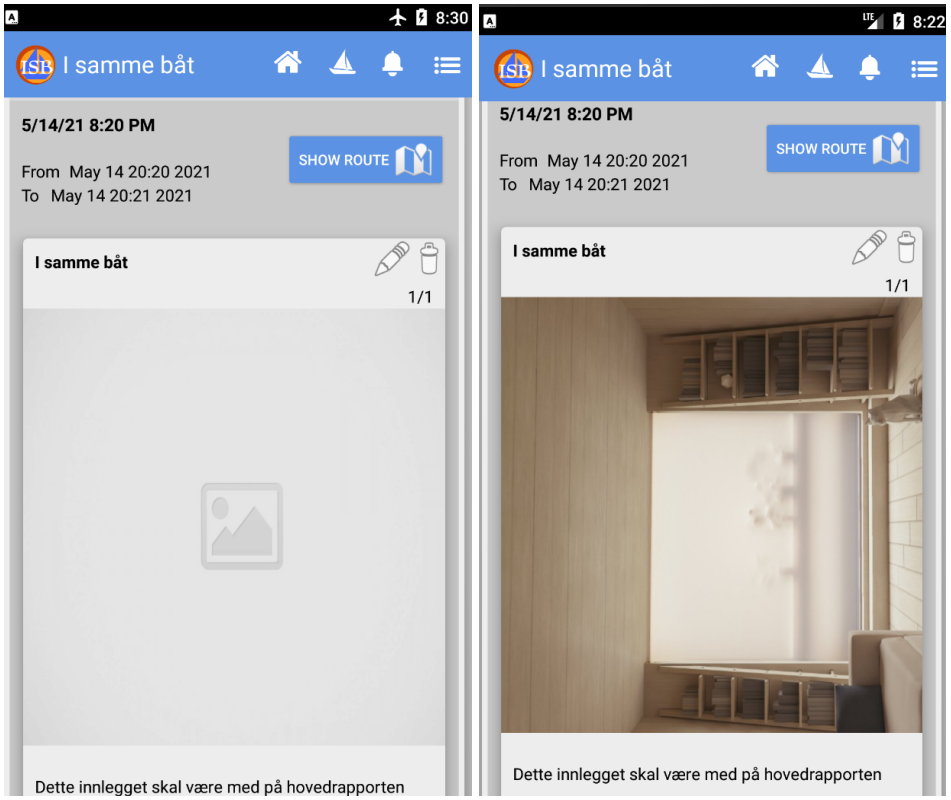
En av hovedfunksjonalitetene ved appen er å kunne spore posisjonen til båten og se hvor man er i forhold til resten av ruta på kartet. Sporingen via GPS er uavhengig av internett, men lagringen og visningen av de tidligere punktene på kartet er problematisk når flere brukere er med på samme tur og man ikke har dekning. Derfor har vi implementert at hver klient som ikke startet turen sporer sine egne punkter hvis de skulle miste nettet og heller sørge for at de laster inn de faktiske punktene sporet av den som startet turen når de igjen oppnår tilgang til internett.

4.1.3 Innloggingssystem

All funksjonalitet i applikasjonen vår er avhengige av tokenet for å autorisere brukere. Derfor var vi nødt til å implementere en innloggingsfunksjon i offline modus. Innloggingssystemet i

offline modus ble implementert ved å kryptere og lagre e-postadressen og passordet i en fil kalt “SharedPreferences” med privat modus. Privat modus betyr at det kun er appen som har tilgang til data som har en nøkkel-ID som utviklere setter selv. Dersom brukerne allerede var innlogget før de mistet dekning, vil de bli logget inn automatisk uansett om de har nettverksforbindelse eller ikke. Da bruker klienten den lagrede innloggingsinformasjon for å logge inn brukeren i bakgrunnen. Hvis brukerne ikke var innlogget blir de sendt til innloggingssiden for å logge inn manuelt, men for å gjøre dette må man vente til man har dekning.

4.1.4 Designet



Figur (4.1): skjermbilde av hjemmeside uten nett

Figur (4.2): skjermbilde av hjemmeside med nett

Designet er en del av produktet vi fokuserte på siden problemstillingen handler om hvordan vi kan ha så lite endring fra når enheten har dekning og ikke, særlig med tanke på brukergrensesnittet. Ordet design inkluderer strukturen i appen, farger og bruk av ikoner. Figur (4.1) viser at vi ikke viser bildene uten dekning, men viser heller et standardbilde hvor det vanligvis ville vært et bilde. Brukeren kan ikke fjerne bilder på logginnlegg uten dekning, men vi setter ingen stopper for at brukeren kan legge til bilder. Dette er i motsetning til online

modus der brukeren kan se og slette bildene. Varsel-ikonet i begge figurene over er synlig, men varselsystemet er ikke tilgjengelig når brukeren ikke har dekning.

4.1.5 Brukertester

Hovedmålet med brukertestene var å teste om brukere merket store forskjeller mellom appen når den var tilkoblet til internett og ikke. På alle testene hvor vi slo av internett uten at testpersonene visste det, ble de alle overrasket da vi avslørte det mot slutten av testen. Sett bort fra de testene der det oppsto noen problemer vi ikke hadde forutsett med koden, var det et par merkelige ting noen brukere merket seg da vi var i offline modus. Når det ikke er nettverksforbindelse blir kun den delen brukeren allerede har sett på lastet inn, mens resten av kartet blir ikke det. En bruker merket dette godt da hen skulle zoome inn på kartet og flere detaljer ikke ble synlig. Brukeren tolket dette som et problem med kartet. Et par brukere prøvde seg på funksjoner som trenger nettilgang og da fikk de beskjed om at de ikke var tilkoblet. På grunn av disse hendelsene var det noen testbrukere som mente det hadde vært lurt å gi beskjed til brukeren i situasjoner hvor det ikke er dekning.

4.2 Ingeniørfaglige resultater

I visjonsdokumentet har vi dokumentert alle funksjonene vi planla å jobbe med i prosjektet. Disse funksjonene går igjen i både produktbackloggen, sprintbackloggene og Gantt-diagrammet. Mange av disse funksjonene har endret seg mye i løpet av prosjektet, noe vi vil diskutere mer i 5.2. Her vil vi beskrive hvilke funksjoner vi har fått implementert, hvordan de fungerer og utdype litt om hva som mangler. En bredere forklaring på hvorfor noen funksjoner ikke har blitt implementert kommer i underkapittel 5.2.

4.2.1 Funksjonelle krav

Nr	Navn	Implementert	Kommentar

1	Registrere bruker	Ja	Vi validerer det brukeren fyller inn og sjekker at passord og mail er på riktig format. Om det ikke stemmer overens med formatet gir vi tilbakemelding om det. Vi sjekker om mailen er registrert fra før og gir tilbakemelding ut fra dette. Passordet blir hashet med salt før det blir lagret. Om registreringen er vellykket får bruker en velkomstmail, bekreftelse i appen og blir sendt til innloggingssiden.
2	Logge inn	Ja	Vi validerer det brukeren fyller inn og sjekker at passord og mail er på riktig format. Om det ikke stemmer overens med formatet gir vi tilbakemelding om det. Om innloggingen blir vellykket lagrer vi mail og passord til neste gang så brukeren slipper å skrive det inn hver gang. Informasjonen blir kryptert. Om bruker har en pågående tur vil hen bli sendt til den, ellers ender hen opp på forsiden av appen.
3	Spore en tur med gps	Ja	Vi sporer enhetens posisjon bare når en tur er pågående. Det blir opprettet et nytt punkt hver 15. meter i ruta. Dette blir vist på et kart med en egen komponent som viser hvor båten befinner seg, samt ruta til turen. Om bruker beveger på kartet vil visningen slutte å følge båten. Kartet følger båten på nytt igjen om brukeren trykker på knappen for dette.
4	Se en tur på sjøkart	Ja	På alle kartvisninger er hovedkartet som vises et sjøkart. Det dekker all territorium underlagt Norge. Utenfor grensene vises et mindre detaljert kart.
5	Lagre reiser med posisjonssporing	Ja	Underveis i turen lagres posisjonene i databasen og når brukeren velger å avslutte turen, setter vi ankomsttidspunktet. Da vil vi slutte å spore posisjonen til enheten og turen vil gå fra pågående til ferdig. I ettertid er det mulig å finne turen i en liste på forsiden av appen.
6	Registrere båt	Ja	Fra båtsiden kan brukeren legge til en ny båt. Om brukeren allerede har en båt fra før er valget litt bortgjemt i en nedtrykksmeny. Her vil man også kunne velge hvilken båt som skal vises og fungere som aktiv mellom alle båtene man er tilknyttet. På registreringssiden kan man fylle inn navn, registreringsnummer og legge til bilde. Navn er obligatorisk for å få opprette en båt. Om brukeren har andre båter fra før vil hen bli spurt om hen vil bruke denne båten som aktiv båt. Uansett blir bruker sendt tilbake til båtsiden etter registrering..

7	Fylle inn overordnet informasjon om turen	Ja	Når en bruker starter tur går vi rett til sporing av enheten og visning på kartet. Informasjonen om turen som tittel, beskrivelse og destinasjon kan fylles inn via redigeringsknappen på turen. Der kan man også legge til passasjerer på turen. Disse kan enten merkes av i en liste over tilknyttede brukere på båten, legges til med navn om de er lagret som gjester på båten eller de kan legges til med mail. All infoen som blir fylt inn vises i detaljer-delen av den pågående turen eller øverst i visningen når turen er ferdig.
8	Knytte tekst og bilde til en posisjon	Ja	I en pågående tur kan man trykke på et ikon nederst på skjermen som tar brukeren til en egen skjerm for å legge til et logginnlegg. Her kan brukeren fylle inn tittel og beskrivelse og så mange bilder som ønskelig fra både kamera og galleri. Bildene kan også fjernes om det ble lagt inn feil. Når logginnlegget blir lagret vises det på kartet i pågående tur på posisjonen brukeren var på da han klikket for å legge til et innlegg.
9	Se alle logginnlegg som hører til en tur	Ja	Når en tur blir lastet inn i klienten vises også alle logginnleggene som har blitt lagt til på turen på sine respektive posisjoner på kartet. Logginnleggene er klikkbare og blir da vist i en popup på skjermen. Det er også en knapp som fører brukeren til en egen side hvor logginnleggene er sortert kronologisk i en liste.
10	Se tidligere turer ut fra en liste	Ja	Alle turene brukeren er knyttet til ligger i en liste på hovedsiden i appen. De ligger organisert under hvilken båt de ble startet på på og har alle logginnleggene som ble lagt ut på turen organisert inni seg. De er sortert i kronologisk rekkefølge basert på avgangstidspunktet. Turene har også en knapp hver som fører brukeren til en visning av ruta til den valgte turen.
11	Se oversikt over båter	Ja	Båter har en egen side hvor vi viser all informasjon om båten som er aktiv for øyeblikket. Den aktive båten er den en tur blir registrert på om brukeren velger å starte en ny tur. Fra båtsiden kan man endre hvilken båt som er aktiv, redigere båten, slette båten, gå til havnvisningen over havnene som er knyttet til båten og legge til og administrere brukere som er tilknyttet til båten.
12	Legge ved brukere og navn som medreisende i en tur	Ja	I redigeringsdelen av en pågående tur kan man legge til passasjerer til båten. Det er bare brukeren som startet turen som har mulighet til det. Brukere tilknyttet til båten som regulær (som vil si nære venner eller familie) vil ligge i en liste hvor brukeren kan merke av hvem

			som er med på turen. Brukere tilknyttet som gjest på båten vil dukke opp som forslag når brukeren skriver inn enten navnet eller mailen deres. Brukere uten noen tilknytning til båten fra før må legges til ved mail. Om brukerne er registrerte i databasen vil de få varsel i appen, ellers vil eieren av mailadressen få en e-post som forteller at hen er invitert på tur og må laste ned appen for å bli med. Brukeren som startet turen kan også fjerne personer som ikke skal være med på turen likevel.
13	Dele en tur med en spesifikk bruker	Ja	Brukeren som startet en tur har mulighet til å dele turen med andre brukere som ikke er om bord. Dette gjøres enten ved å skrive inn navn eller mail på en bruker tilknyttet som gjest på båten, eller ved å skrive inn mail på en bruker uten noen tilknytning. Regulære brukere på båten og admin-brukere dukker ikke opp som forslag og kan ikke legges ved som navn da de allerede har lesetilgang til alle turene på båten. Brukeren som får turen delt med seg får et varsel i appen om hen er registrert, og ellers en mail som inviterer personen til å registrere seg i appen. Turen vil da dukke opp på hovedsiden for disse brukerne og de kan se ruten enten via listingen der eller via selve varselet de fikk.
14	Kunne få hjelp om man ikke forstår appens oppsett	Ja	Vi gir brukeren hjelp til funksjoner ved hjelp av både informasjonsikoner der det trengs i appen og i en egen hjelpeside som inneholder svar på ofte stilte spørsmål. Hjelpesiden inneholder forklaringer på konsepter i appen som kan være forvirrende. Svarene inneholder både tekst og bilder som illustrerer hvordan funksjonene fungerer.
15	Se andre båters turer	Ja	Brukere som ikke er med som passasjer på en tur kan likevel se turen om hen er regulær eller admin på båten eller om turen har blitt delt med brukeren. Brukeren vil da ha tilgang til en visning av ruta på kart der båten vises i sanntid. Hele ruta vil også være synlig i ettertid når turen er ferdigstilt. Om brukeren bare har lesetilgang, vil alle funksjonene som redigerer noe som helst være deaktivert. Brukeren kan heller ikke legge til logginnlegg, andre brukere som passasjerer eller dele turen videre.
16	Se farten til båten i sanntid	Ja	Når en bruker er inne på siden for pågående tur vil en ekstra service-klasse kjøres som leser posisjonen til enheten raskere enn den vanligvis ville gjort. Hvert sekund sjekker vi posisjonen og

			kalkulerer avstanden fra forrige punkt for så å kalkulere farten. Denne blir fortløpende oppdatert i visningen for brukeren. Måleenheten for farten er vist i knop.
17	Se distansen til turen underveis	Ja	Hver gang en tur får en ny posisjon lagt til i ruta blir distansen oppdatert. Appen regner ut lengden mellom alle posisjonene og legger det sammen. Svaret blir vist i detaljer-delen av siden på pågående tur og er målt i nautiske mil.
18	Skrive logginnlegg til en tidligere passert posisjon	Ja	Brukeren kan velge å skrive et logginnlegg til en posisjon båten har vært innom. Hen velger da en posisjon på ruta og får opp en hjelpeskjerm som spør om bruker vil legge til et logginnlegg her. Da blir brukeren sendt til en egen skjerm for å legge til logginnlegg. I dette tilfelle vil ikke brukeren kunne endre tidspunktet da det blir satt til å være tidspunktet som var da båten befant seg i den valgte posisjonen.
19	Skrive logginnlegg til et tidligere tidspunkt	Ja	I siden for å legge til logginnlegg kan brukeren velge et tidligere tidspunkt. Når logginnlegget skal vises på kartet vil den legges til posisjonen som har det nærmeste tidspunktet til den valgte tiden. Vi kontrollerer og gir tilbakemelding om tidspunktet er utenfor rammene for turen.
20	Få hjelp til å komme i gang	Ja	I en fil for appen kalt SharedPreferences lagrer vi om brukeren har åpnet appen før. Om det ikke har skjedd viser vi en hjelpeskjerm som ønsker brukeren velkommen og forteller hva som må gjøres for å komme i gang. I tillegg vises det på hovedskjermen en hjelpeboks for brukere som ikke har noen tilknyttede båter med en forklarende tekst og en knapp som fører til siden hvor man kan registrere en båt.
21	Redigere informasjon om tur	Ja	Informasjonen om pågående tur kan endres av brukeren som startet turen. Her vil det også være mulig å legge til flere brukere som passasjerer på turen eller fjerne noen av dem. Hvordan dette fungerer er beskrevet i funksjon nr. 7. Den oppdaterte informasjonen vil bli vist i detaljer delen av pågående tur.
22	Starte sporing av tur etter at selve turen har startet, og likevel få med hele turen i loggen	Ja	Om en bruker glemte å starte sporingen av turen før turen var i gang kan man i siden for redigering av tur trykke på en knapp for å sette avgangspunkt. Da vil en ny hjelpeskjerm dukke opp hvor bruker kan velge posisjon på et kart og sette tidspunkt. Om enten tidspunkt eller posisjon ikke blir valgt får brukeren tilbakemelding om det. I tillegg er det en markeringsboks som spør om brukeren vil overskrive

			posisjonen som tidligere var startpunktet for turen. Dette kan skje om brukeren valgte feil posisjon først som avgang, og ville endre det. Den valgte posisjonen blir lagt til i ruta som vises på hovedkartet og det blir trukket strek til den fra neste punkt.
23	Manuelt bestemme endeposisjon og endetidspunkt for en tur	Ja	Om en bruker har glemt å stoppe sporingen av turen ved ankomst kan man velge å sette en tidligere posisjon som ankomstpunkt. Når brukeren velger å avslutte turen, vil en hjelpeskjerm dukke opp som spør om turen allerede har kommet til sin endeposisjon. Denne linken vil åpne enda en hjelpeskjerm med et kart som viser alle posisjonene i ruta til turen. Brukeren kan da velge en tidligere posisjon eller et tidligere tidspunkt for når turen egentlig var over. Om brukeren velger tidspunkt blir den nærmeste posisjonen automatisk valgt og vice versa. Posisjoner etter den valgte blir da fjernet etter brukeren godtar en advarselsskjerm.
24	Arkivere flere turer som en del av samme reise	Nei	Denne funksjonen er ikke implementert. Vi har grunnmuren i databasen med en egen tabell med en fremmednøkkel i tur-tabellen, men utenom det er ingenting utviklet.
25	Legge til havn på kartet manuelt	Ja	En bruker kan fra havnesiden velge å registrere en havn manuelt, i motsetning til de som blir lagret automatisk ved ankomst av en tur. Den er tilgjengelig fra båtsiden og viser havnene som allerede er tilknyttet båten på et eget kart. Bruker må fylle inn navn på havn og sette en posisjon på kartet. Om det ikke blir gjort får brukeren tilbakemelding om det.
26	Legge til havn automatisk på avgangspunkt og ankomstpunkt etter en tur	Ja	Når en tur blir avsluttet sjekkes posisjonene ved ankomst og avgang. Om de ikke er i innenfor en 50 meters radius til en eksisterende havn vil det bli registrert en ny havn på denne posisjonen. Navnet på havnen blir først satt til “ukjent”, men neste gang bruker avslutter turen der vil hen få valget om å endre navn på den. Om brukeren ikke har endret navnet på turen skal det automatisk bli endret til “Fra [starthavn] til [slutthavn]”.
27	Redigere både navn og posisjon til en eksisterende havn	Ja	Fra havnesiden kan brukeren trykke på havnene som vises på kartet og trykke videre til redigering av havn ved hjelp av et redigeringsikon. Da vil en hjelpeskjerm dukke opp med et nytt kart hvor brukeren kan velge ny posisjon og redigere navn på havnen. Om navnefeltet har blitt tømt får brukeren feilmelding om dette. Havnen vil bli satt til den nye posisjon på havnesiden og vist med

			nytt navn. Turer som har gått til denne havnen vil få navnet sitt oppdatert. I tilfelle havnen har endret posisjon må vi endre tittelen på turen til "ukjent" hvor havnenavnet brukes, men om havnen derimot bare har endret navn må vi endre havnenavnet i tittelen på turen til det nye navnet.
28	Slette en havn	Ja	Fra havnesiden kan brukeren trykke på havnene som vises på kartet og derfra slette havnen ved å trykke på søppelikonet. En advarsel vil dukke opp og forsikre om brukeren er sikker. På kartet vil havnen bli fjernet om brukeren godtar. Turer som har gått til denne havnen vil få tittelen sin oppdatert til "ukjent" hvor havnenavnet brukes.
29	Se havner på kart	Ja	Alle havner tilknyttet en båt blir vist på kartet både under havnesiden og på siden for pågående tur.
30	Tilbakestille passord om glemt	Ja	Fra innloggingssiden kan brukeren gå til en "glemt passord"-side. Her er det mulig å fylle inn mail for å få tilsendt en midlertidig kode. Vi sjekker mailen og gir tilbakemelding om den er i riktig format og er registrert i databasen. Koden består av fire tegn som enten er store bokstaver eller tall og er gyldig i et kvarter. Om brukeren fyller inn riktig kode vil hen bli sendt til en egen side hvor nytt passord må fylles inn to ganger for å forsikre at passordet blir riktig.
31	Redigere logginnlegg	Ja	Brukeren kan redigere logginnlegg fra hovedsiden, fra logginnlegg-feeden, og fra kartet i både pågående tur og tidligere tur. Det eneste stedet et logginnlegg vises og det ikke er mulig å redigere det er hvor man søker etter logginnlegg i nærheten. Redigeringsknappen vises kun for admin-brukere og den som har skrevet innlegget. Knappen tar brukeren til en ny skjerm hvor hen kan endre tittel og beskrivelse på logginnlegget. I tillegg kan man fjerne bilder eller legge til nye. Etter at brukeren trykker på "lagre" skal hen bli sendt tilbake til forrige skjerm.
32	Slette logginnlegg	Ja	Samme regler for visning av knappen gjelder som for redigering av logginnlegg. Ved å trykke på sletteikonet får brukeren opp et varsel som spør om hen er sikker. Om bruker bekrefter blir logginnlegget fjernet i databasen og i visningen til brukeren. Bilder som er koblet til logginnlegget i databasen blir også fjernet.
33	Få varsel	Ja	Når brukeren har logget inn, vil det alltid være en verktøylinje øverst på skjermen. Der vises et bjelleikon som åpner en liste med varsler når det blir trykket på. Om brukeren får et nytt varsel vil ikonet få et

			rødt merke på seg med utropstegn. Når brukeren har trykket på ikonet for å se varslene, blir merket borte. Appen kjører en forespørsel til serveren etter nye varsler hvert 30. sekund såfremt appen kjører.
34	Bekrefte å bli lagt til på en tur eller båt	Delvis	Når en bruker blir lagt til på en tur eller båt, får brukeren et varsel om det. Her er det to valg ved enten å godta eller avslå invitasjonen. Om brukeren godtar skjer det ingenting og brukeren fortsetter å være på turen eller båten, men om brukeren avslår blir brukeren fjernet fra turen eller båten. Det vi ikke har fått implementert er at turen eller båten ikke skal vises for brukeren før hen har godtatt invitasjonen, i tillegg til at navnet på brukeren skal vises som "ikke-akseptert" eller noe lignende for brukere som kan se en visning av turen eller båten.
35	Redigere logginnlegg man ikke har skrevet om man er admin	Ja	Brukere med admin-rolle på en båt kan redigere alle logginnlegg som er knyttet til denne båten. Brukere som ikke er admin på båten kan kun redigere sine egne logginnlegg.
36	Slette logginnlegg man ikke har skrevet om man er admin	Ja	Brukere med admin-rolle på en båt kan slette alle logginnlegg som er knyttet til denne båten. Brukere som ikke er admin på båten kan kun slette sine egne logginnlegg.
37	Slette tur	Ja	Brukere kan bare slette en tur om de startet turen eller er en admin-bruker på båten. Da vil sletteikonet vises i siden for tidligere turer. Når ikonet trykkes på, vil et bekreftelsesvarsel dukke opp. Om brukeren godtar vil turen samt alle tilhørende logginnlegg og bilder slettes i databasen og fjernes fra visningen til brukeren. Brukeren vil da bli sendt tilbake til forrige side.
38	Slette båt	Ja	Brukere kan bare slette en båt om de er registrert som admin på båten. Da vil sletteikonet vises i båtsiden. Når det trykkes på, vil et bekreftelsesvarsel dukke opp. Om brukeren godtar vil båten slettes i databasen med alle tilhørende turer, logginnlegg og bilder, og den fjernes fra visningen til brukeren. Siden vil oppdateres og aktiv båt vil endres til den neste båten i listen. Om brukeren ikke har flere båter vil siden bli tom med instrukser for hvordan en ny båt kan registreres. Det trengs ingen godkjenning fra andre adminbrukere på båten før båten blir slettet.

39	Søke etter logginnlegg i nærheten	Ja	I pågående tur kan en bruker søke etter logginnlegg i nærheten av båten ved å trykke på et forstørrelsesglassikon i detaljer-seksjonen. I et hjelpevindu vil det dukke opp et kart og et felt for å fylle inn radien man vil søke i. Dette søker etter alle logginnlegg brukeren har tilgang til som har blitt skrevet i nærheten og viser det på et kart som tilpasser seg for å vise alle innleggene. Herfra kan brukeren også gå inn på tilhørende tur til hvert av logginnleggene.
40	Se andres profiler	Nei	Profil for brukeren selv eller visning av andres profiler er ikke implementert.
41	Redigere brukerinformasjon	Ja	Fra "mer"-valget i verktøylinja kan brukeren velge å redigere brukerinformasjonen sin. Alt kan redigeres, som navn, mail og også legge ved profilbilde (selv om profilbilde ikke har noen funksjon i appen til dags dato). Tilslutt må bruker bekrefte endringene med passordet sitt. Vi sjekker om mail er riktig format, at ingen av feltene er tomme og om passordet er riktig. Om mailen er endret til en adresse som allerede ligger i databasen, får brukeren feilmelding om dette.
42	Endre passord		Fra "mer"-valget i verktøylinja kan brukeren velge å endre passordet sitt. Brukeren må fylle inn sitt nåværende passord, samt et nytt passord to ganger for å bekrefte. Om noen av feltene er tomme, det gamle passordet ikke stemmer eller de to nye passordene ikke stemmer overens avbryter vi og brukeren får tilbakemelding på feilen. Om alt går gjennom vil nytt passord hashes med salt før det lagres i databasen.
43	Redigere båt	Ja	På båtsiden kan brukere som er satt til admin på båten redigere båten ved å trykke på et redigeringsikon. Det åpnes da en ny side hvor brukeren kan endre båtnavn, registreringsnummer og bilde. Om navnefeltet blir stående tomt gir vi tilbakemelding på det. Det er bare navn som er obligatorisk å fylle inn. Etter fullført redigering vil brukeren bli sendt tilbake til båtvisningen hvor informasjonen er oppdatert.
44	Legge til tilknyttede til båt	Ja	Fra båtsiden kan en adminbruker legge til brukere som tilknyttede ved båten. Det kan gjøres via mail. Om mailadressen ikke finnes i databasen vår sender vi en mail til adressen og forteller at adminbrukeren har lagt hen til i båten og inviterer til å laste ned appen. Om mailadressen derimot finnes i appen får denne brukeren

			varsel inni appen om situasjonen. En tredje måte å legge til brukere på er å ikke skrive mailadresse, men bare fylle inn navn. Da vil det opprettes en "uregistrert bruker" som tilhører båten. Dette kan være for barn eller andre som ikke vil at mailadressen deres skal lagres i systemet. En uregistrert bruker kan senere bli redigert av adminbrukere til å få mailadresse eller endre navn. Man kan ikke legge til brukere via mail om en bruker med mailadressen allerede er tilknyttet båten. Vi sjekker også om formatet på mailen er riktig og gir tilbakemelding om det.
45	Kunne tilknytte gammel informasjon om ditt navn til din bruker	Ja	Om en mailadresse allerede er tilknyttet en eller flere båter når en bruker registrerer seg vil det dukke opp et varsel om dette. Her vil bruker få muligheten til å skrive inn navnet på båten(e) hen vil opprettholde tilknytningen med. Koblingene til båtene brukeren ikke skriver inn blir fjernet ved registrering. Koblingene bruker velger å opprettholde vil dukke opp når brukeren logger inn i tillegg til at varsler blir opprettet som opplyser om alle koblingene.
46	Legge til logginnlegg som tilhører en båt	Ja	Fra hovedsiden kan en bruker legge til et logginnlegg som ikke tilhører en tur, men bare er knyttet direkte til en båt. Knappen vises bare om brukeren ikke har en pågående tur og om brukeren har en aktiv båt. Når brukeren trykker på knappen blir hen sendt til den vanlige siden for å skrive logginnlegg. Ved lagring vises innlegget kategorisert under båten på hovedsiden.
47	Søke gjennom tidligere turer	Ja	Denne funksjonen har blitt utvidet og endret til å inkludere logginnlegg. Nå søkes det på tittel og beskrivelse både på logginnleggene som ligger under tur og de som ligger direkte under båt. De båtene og turene som ikke har treff på sine logginnlegg blir ikke vist. Søkingen skjer på hovedsiden og her kan også brukeren filtrere visningen etter båt.
48	Bytte mellom ulike typer kart	Nei	Denne funksjonen er ikke implementert.
49	"Like" en logg	Nei	Denne funksjonen er ikke implementert.
50	Slette konto	Ja	Fra "mer"-valget i verktøylinja kan brukeren velge å slette kontoen sin. Når brukeren trykker på knappen, vil en bekreftelsesboks dukke opp. Om brukeren bekrefter vil all brukerinformasjon bli slettet fra databasen. Om brukeren er den siste adminbrukeren på en båt, vil

			båten(e) bli slettet sammen med alle turene og logginnleggene. Brukeren vil få et ekstra varsel som forklarer dette. Om bruker derimot ikke er siste adminbruker på en båt vil båten fortsette å eksistere og turene og logginnleggene brukeren har skrevet fortsatt finnes i databasen. Brukeren blir så sendt til innloggingssiden.
51	Kommentere en logg	Nei	Denne funksjonen er ikke implementert.
52	Logge inn med bruker fra andre sosiale medier som facebook eller google	Nei	Denne funksjonen er ikke implementert.
53	Administrere hvem som kan se din posisjon	Nei	Denne funksjonen er ikke implementert.
54	Bli varslet om brukere i nærheten	Nei	Denne funksjonen er ikke implementert.
55	Se alt fra applikasjonen på en tilknyttet nettside	Nei	Denne funksjonen er ikke implementert.
56	Spore forbruk (vann, olje, diesel)	Nei	Denne funksjonen er ikke implementert.

Figur (4.3): Status på funksjoner

4.2.2 Ikke-funksjonelle krav

Kapittel 6 i visjonsdokumentet definerer ikke-funksjonelle krav vi satte til produktet i planleggingsfasen. Kravene var som følger:

- *Alle data som benyttes av applikasjonen "I samme båt" skal lagres i database. Studentene og oppdragsgiver har avtalt at NTNUs mySQL-database skal brukes.*

I utviklingsfasen av produktet brukte vi NTNUs MySQL-database, men for å sette opp en server vi kunne koble opp mot når vi ikke kjørte på en emulator endte vi opp med å bruke Google Cloud-systemet og deres versjon av MySQL-database. Dette er beskrevet mer grundig i kapittel 3.

- *Løsningen skal ha en god brukskvalitet, lett og intuitiv å bruke.*

Vi har strevet for at all funksjonalitet skal være enkel å forstå og lett å bruke. Om dette er noe vi har oppnådd kommer vi tilbake til når vi skal diskutere resultater fra brukertester i underkapitlene 5.1.5 og 5.2.5.

- *Både server og klient skal ha gode loggsystemer hvor man kan få oversikt over hvilke koblinger som har blitt oppført og hvilke funksjoner som har blitt utført samt gode beskrivelser av eventuelle feil som kunne oppstå.*

Dette kravet har vi delvis implementert. Loggsystemet er på plass i serversiden og brukes etter ønske fra produkteieren der det er viktig å fange feil med mer detaljer og ikke bare utskrifter. Til å sette opp et loggsystem brukte vi rammeverket "SLF4J" som gir tilbakemelding og hjelper utviklere å feilsøke i koden. Loggsystemet muliggjør også å skrive ut i terminalen under forskjellige kategorier, blant annet informasjon, advarsler, debug og feilmeldinger.

- *Koden skal ha svært stor testdekning med både enhetstesting og integrasjonstesting med spesielt vekt på sistnevnte. Ingen spesifikk prosentandel er nevnt, men ca 90-95% ligger underforstått.*

Dette kravet har vi ikke fått oppfylt til ønsket grad. Det tok lang tid før vi fikk kommet ordentlig i gang med servertesting, og enda lengre tid før vi fikk startet på klienttesting. Oppsettet på spesielt sistnevnte var mer komplisert enn først forventet og tok av den grunn lengre tid enn forventet å komme i gang med. På ett tidspunkt hadde vi opp mot 80% dekning av serveren, men så ble det lagt til noen nye metoder vi ikke fikk tid til å teste. Vi endte opp med en dekning av endepunktene i serveren på 70%. Klienttestingen var mye vanskeligere å jobbe med enn forventet og resultatene mellom hver kjøring var veldig inkonsistente som førte til at det var vanskelig å kjøre testene og få et resultat.

- *Applikasjonen skal ikke lagre for mye informasjon av brukeren. Kun nødvendige detaljer, som i første utkast bare vil bestå av epost og navn. Om bruker vil dele sin posisjon med andre, må brukeren få klar beskjed om hva dette innebærer.*

Vi har valgt å ikke utvide mengden informasjon som lagres om brukeren. For å registrere en bruker må man fylle inn navn og mail. Senere kan man også legge inn et profilbilde, men det er ikke et krav og har heller ingen funksjon slik appen er bygd opp nå. For å styrke personvernet for barn og andre ønskede har vi en mulighet for å bare legge inn navn på en ikke-registrert bruker. Denne kan oppgraderes til en fullverdig bruker, men kan også bare ligge som et navn for å koble mot turer.

Vi har også vært påpasselige om at all data som blir lagt inn enkelt skal kunne fjernes om ønskelig. Logginnlegg, turer og til og med båter er enkelt å slette om man ikke ønsker at dataen om disse skal ligge i appen. Funksjonen for å slette brukerkontoer er også implementert. Disse slette-funksjonene skal påvirke hverandre på den måten at om en tur blir slettet skal alle logginnleggene på turen også bli slettet. Det samme gjelder også for båter når bruker blir slettet, dersom brukeren er den siste adminbrukeren på en båt. Da vil båten, alle dens logginnlegg og turer bli slettet.

4.2.3 Testresultater

Vi kjørte i alt 13 brukertester i tillegg til at produkteier i flere omganger testet appen og ga oss tilbakemeldinger. Vi lagde tre ulike sett med tester for å teste ulike aspekter ved appen uten at en test skulle ta altfor lang tid. Da vi hadde fått gode tilbakemeldinger der og hadde rettet opp i noen enkle designendringer la vi til varianten at vi skulle slå av internett underveis i testen uten å fortelle brukeren om det. Det første settet med oppgaver skulle teste de grunnleggende funksjonene i appen med bruker- og båt-registrering, starte en tur og skrive og redigere et logginnlegg. Den andre testen skulle teste noen mer avanserte funksjoner for å avdekke om de var intuitive nok ved hjelp av designet. Den tredje testen skulle teste noen flere avanserte funksjoner i tillegg til at vi hadde satt opp noen koblinger til brukerkontoen som ble brukt for å teste det sosiale aspektet mellom brukere. En stadig gjenganger i testene var at vi oppdaget nye feil i systemet vi ikke hadde merket før.

Vi fikk mange tilbakemeldinger på design, hvor de aller fleste i hovedsak var positive. Noen mente fargevalgene våre var fine, mens andre mente fargene var litt kjedelige. Mange slet med å komme seg til hovedsiden hvor alle logginnleggene er og hvor man starter en ny tur. De skjønte ikke at man skulle trykke på logoen eller navnet på appen for å komme seg dit. Et annet problem flere hadde med navigasjonen var at de syns det var vanskelig å komme seg tilbake uten en dedikert tilbakeknapp på siden til tross for at Android-telefoner har en egen tilbakeknapp innebygd i designet sitt. Flere slet med å finne hvor man kunne søke etter logginnlegg i nærheten og mente at forstørrelsesglassikonet ikke var beskrivende nok til den funksjonen. Lignende tilbakemeldinger fikk vi angående knappen for å legge til et nytt logginnlegg, mens andre forstod at det måtte være det knappen skulle brukes til. Et par av testbrukerne forstod heller ikke helt hva "Mine båter"-knappen i verktøylinja betydde. Til tross for et par problemer med navigasjon mente de fleste at oppsettet var godt strukturert, logisk og var enkelt å forstå ut ifra erfaringene de hadde fra andre apper.

Flere av testbrukerne prøvde flere ganger å med vilje å finne mangler ved produktet. Noen skrev inn uvanlige lange tekster i innskrivingsfelter for navn for å se hva som ville skje. Dette hadde vi ikke forutsett og førte til noen uventede utfall. Vi oppdaget også at vi ikke hadde stor bokstav som en standard i innskrivingsfelter. I tillegg fikk vi tilbakemeldinger om at det var flere situasjoner hvor brukeren ikke var helt klar over hva som skjedde og gjerne skulle fått mer tilbakemelding. Dette dukket for eksempel opp ved registrering av bruker hvor man ikke fikk noen annen tilbakemelding enn å bli sendt til innloggingsskjermen. Flere var også ikke helt sikre på om turen faktisk hadde startet da de trykket på "start tur" siden appen ikke ga noen spesifikk beskjed om det.

4.2.4 Produktstatus ved innlevering

Satt opp mot produktet som beskrives i visjonsdokumentet er produktet vi har produsert tilstrekkelig i noen aspekter, ikke helt tilstrekkelig i andre og forbigår noen aspekter med store utvidelser utenfor grensene til de originale funksjonene. Bekreftelse av tilknytning er ikke implementert til sitt fulle potensial, reiser er nesten ikke jobbet med og andre mindre funksjoner mangler. Mange funksjonaliteter som var planlagt som en del av produktet har blitt endret til noe annet, mens noen står igjen som mulige utvidelser ved videre utvikling. For eksempel er profilvisning, meldingssystem og endring av kartvisning er funksjonaliteter

som ikke er ferdig utviklet. Derimot har vi utvidet konseptet med tilknytning av bruker gjennom båter og turer og utviklet systemer med roller og rettighetsnivå på hver av dem. Hovedfunksjonaliteten til appen er på plass og i tillegg har vi oppfylt et stort krav fra produkteier med at appen fungerer selv om enheten ikke har dekning.

4.3 Administrative resultater

Som tidligere nevnt brukte vi Scrum som utviklingsmetodikk. Hver sprint varte i mellom 7 og 11 arbeidsdager, og vi satte mål vi skulle prøve å oppnå i hver sprint. Nesten hver sprint hadde som overordnet mål å ha implementert et nytt aspekt ved applikasjonen, og disse målene ble satt opp i backloggen til den tilhørende sprinten. Sprint 1 hadde som hovedmål å ha implementert et fungerende registrerings- og innloggingssystem. Sprint 2 hadde som hovedmål å kunne registrere en båt, og å kunne starte og avslutte en tur på denne båten. I sprint 3 var fokuset på å ha implementert all hovedfunksjonalitet knyttet til logginnlegg. Sprint 4 hadde fokus på det sosiale aspektet ved applikasjonen, inkludert å tilknytte flere brukere til båter, turer og hverandre. Sprint 5, som var den siste fullverdige sprinten hvor utvikling var hovedfokuset med arbeidet, ble brukt til å implementere tilleggsfunksjoner (bl.a. havner), testing av kode og å fikse feil med tidligere funksjoner.

Under er en oversikt over alle sprintmålene vi hadde, og hvorvidt de ble oppnådd eller ikke:

Sprint	**Målene som ble oppnådd**	**Mål som ikke ble oppnådd**
1	- Å ha et fungerende registrerings- og innloggingssystem	- Å kunne se posisjonen sin på et sjøkart i applikasjonen
2	- Å kunne se posisjon på kart - Å kunne starte og avslutte en tur - Å kunne registrere en båt	- Testing av bruker-entiteten
3	- Hovedfunksjonalitet over logginnlegg - Å ha et designkonsept og en strukturplan	- Å implementere sjøkart

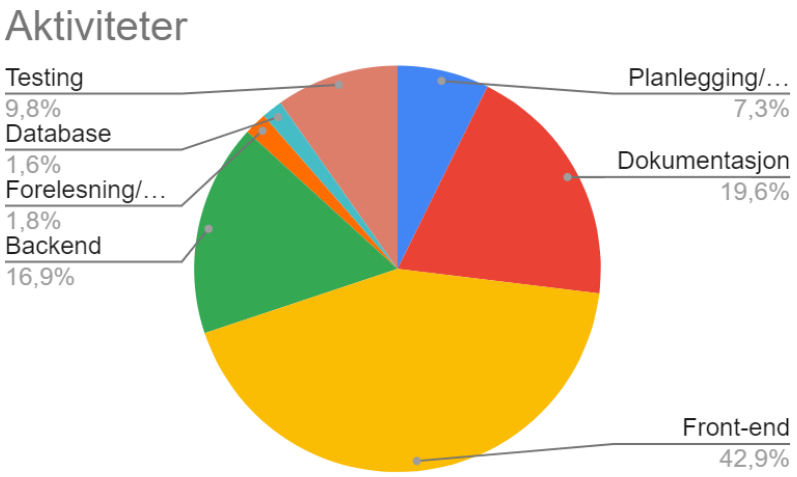
	- Å se egen profil (som senere i prosjektet ble endret til en side med oversikt over brukerens båter)	
4	- Hovedfunksjonalitet ved tilknytning av brukere - Sette opp begrensninger mtp. tilknytning til båt - Å kunne slette båt - Hovedfunksjonalitet ved varsler og deling av tur	
5	- Testing av backend og CI-testing	- Å bli ferdige med all funksjonaliteten til appen

Figur (4.3): oversikt over sprint-målene

Vi brukte også Gantt-diagram som hjalp oss å se fremgangen relatert til prosjektet over tid. Ved starten av prosjektet satte vi opp hvilke funksjoner som skulle gjøres i hver enkelt sprint. Vi valgte også å oppdatere Gantt-diagrammet ved hver sprintstart for å estimere mer nøyaktig når i sprinten vi tenkte å jobbe med det og omtrent hvor mange dager vi ville bruke. Samtidig valgte vi å ikke endre på tidligere estimater for å gjenspeile virkeligheten, men heller la det stå som historikk over planleggingen. Derfor vil det se ut som at noen funksjoner har blitt jobbet med veldig lenge og over flere sprinter. Dette gjelder blant annet funksjonen med sjøkart som vi planla å begynne med i sprint 1, men ikke fikk startet med (se Gantt-diagrammet i *Prosjekthåndboka*). I Gantt-diagrammet ser det da ut som vi har jobbet med det i store deler av sprint 1. Diagrammet har hjulpet oss til å holde styr på funksjoner som ikke ble ferdig implementert til en satt tid. Den viser også hvordan prosjektet har endret seg underveis i prosessen og at vi ikke alltid hadde rett i estimatene våre.

For å dokumentere fremgangen til prosjektet brukte vi et timeregnskap, sprintbacklogg, og møteinnkallinger og tilhørende referater. I timeregnskapet delte vi arbeidet inn i syv

forskjellige kategorier: planlegging/møter, dokumentasjon, front-end, back-end, forelesning/workshop, database og testing. Dette er den sammenlagte fordelingen av arbeid på aktiviteter i prosjektet:



Figur (4.4): Oversikt over fordeling av arbeid på aktiviteter

Over to femtedeler av prosjektet ble altså brukt på front-end, dvs. utvikling av selve applikasjonen. Omtrent en sjettedel ble brukt på back-end, som vil si server-delen av systemet. Nesten en femtedel ble brukt på dokumentasjon. Omtrent en tiendedel har blitt brukt til testing. Denne kategorien omfatter både testing av kode og brukertester. Litt under en tiendedel har blitt brukt til planlegging og møter. De resterende 3.4 prosentene har blitt brukt til arbeid med databasen, obligatoriske forelesninger og workshops vi har hatt i faget. Det at database-kategorien bare har 1.6 % er litt misvisende, da enkelte teammedlemmer har kategorisert noe av arbeidet med database i backend, siden disse to kategoriene er noe overlappende.

Vi hadde et nært samarbeid med produkteier i løpet av prosjektet. Vi hadde vanligvis ukentlige møter, og ellers hyppig mailkorrespondanse når vi var usikre på noe eller trengte hjelp. I enkelte tilfeller hadde vi også sesjoner hvor vi fikk hjelp til å konfigurere programvare, for eksempel Google Cloud-serveren og integrasjonstesting. Vi hadde sprintavslutning med både produkteier og veileder omtrent annenhver uke hvor vi fikk tilbakemelding på det som var gjort. Under sprintavslutningene vurderte vi også om vi hadde klart å oppnå målene vi hadde satt for sprinten.

Kapittel 5: Diskusjon

I dette kapitlet skal vi diskutere og vurdere data og resultater beskrevet i kapittel 4. Derfor har dette kapittelet samme struktur som kapittel 4. I tillegg skal vi gå gjennom hva som var bra, hva som kunne bli gjort annerledes, svakheter og tilslutt drøfte arbeidet med tanke på det helhetlige systemet.

5.1 Vitenskapelig diskusjon

Vår problemstilling omhandler brukergrensesnitt og hvordan vi kan opprettholde en god brukeropplevelse når nettverksforbindelsen er sporadisk dårlig. For å kunne jobbe med dette måtte vi først ha et robust system på plass som fungerer godt når man har god dekning. Mye av prosjekttiden har gått med på å bygge opp denne grunnmuren og sørge for at vi har et godt grunnlag å jobbe videre med når vi skulle implementere at appen skulle fungere uten internett. Oppkoblingen fra klienten til serveren går via HTTPS-forespørsler og responderende svar. Under en vanlig kjøring av appen hvor enheten har dekning vil klienten sende forespørsler om data å vise til brukeren eller forespørsler med endringer som brukeren har utført.

5.1.1 Lokal lagring

Det er mange fallgruver når man skal bygge opp en app med en tilgjort kobling mot en server og det er mange situasjoner hvor ting kan gå galt. Et tilfelle vi forutså som kunne bli et problem er at vi først bare lagret data fra GET-forespørsler etter at brukeren har bedt om det selv, men om brukerne vil se data de ikke har sett tidligere som for eksempel en tur de har blitt delt med, vil ikke denne være tilgjengelig uten nett. For å forhindre denne mangelen på informasjon kjører vi alle de mulige GET-forespørslene vi vil trenge når brukeren logger inn og overskriver data som ligger i minnet fra før. Dette er også for å oppdatere filene om de har blitt utdaterte ved at andre brukere har endret noe på dataen. På den andre siden kan det ha en negativ påvirkning på responstiden ved innlogging, siden det er mye data som må hentes.

Under resultater har vi nevnt at vi lagrer alle GET-responser fra server for å vise dem til brukerne når de mister nettet. Det vil ikke alltid stemme om noe har forandret seg i mellomtiden. For å forhindre at de lagrede responsene blir utdaterte, sørger vi for å oppdatere dem når brukeren endrer noe i klienten. For eksempel ved PUT-forespørsler lagrer vi som tidligere nevnt forespørselen i en egen fil, men i tillegg må vi lete gjennom filene og se om vi har en tilhørende GET-respons lagret som må endres på. Denne endringen vil skje på serveren når klienten får dekning og kan sende PUT-forespørselen, men her later vi som at det har skjedd og utfører endringene på den lagrede GET-responsen i mellomtiden.

Alle filer vi lagrer har navn som starter med brukeren sin ID. Slik sørger vi for at vi ikke mister noe data selv om en annen bruker logger inn, siden vi skiller mellom filene til hver enkelt bruker. Samtidig ville det oppstått problemer om vi hadde kjørt forespørsler for en annen bruker enn den som er logget inn, da vi regner med at forespørslene som sendes inn hører til brukeren som eier tokenet som sendes ved alle forespørsler. Om dataene ikke stemmer overens med brukerens ID vil vi få feilmelding fra serveren og dataen vil bli tapt. Derfor kjører vi ikke lagrede forespørsler for andre enn brukeren som er innlogget etter at enheten har fått nettverksforbindelse.

Problemstillingen vår går ut på at det skal være så liten endring fra online modus til offline modus, men det finnes noen deler av appen vi må begrense når man ikke har dekning. Det gjelder i situasjoner der vi må dobbeltsjekke det brukeren skriver inn ut ifra hva som ligger i databasen. For eksempel når en bruker skal registrere seg eller legge til mail på en uregistrert bruker tilknyttet en båt. Uten dekning er det ingen måte å sjekke om mailadressen som skal brukes er i bruk allerede, derfor må vi deaktivere disse funksjonene i offline modus. I det store bildet er det en liten del av appen som må deaktiveres på grunn av dette.

5.1.2 Posisjonssporing

Det er alltid bare én klient som faktisk sporer posisjonene som lagres på turen, og det er den som startet turen. Oppdatering av ruta med nye punkter for denne klienten vil fungere på samme måte som forklart tidligere ved at oppdateringsforespørslene blir lagret i en egen fil som blir sendt når man får dekning. Dette fører til problemer for andre brukere som er med på

turen og venter på posisjoner fra serveren. Derfor endte vi opp med å implementere at alle brukere som er med på turen sporer posisjon når det ikke er dekning. På denne måten blir ruta vist for alle som er med på turen. Når brukeren som har startet turen får nett, blir ruten hens lagret i databasen og sendt til de andre medreisende. Da vil posisjonene som de andre medreisende sporet bli overskrevet av den reelle ruta. Forskjellen på den reelle ruta og de som ble sporet av medreisende vil være omtrent like dersom alle er med i samme båt. Det vi ikke kunne implementere er å vise ruten av turen til brukere som turen blir delt med fordi de ikke er med på turen. De vil se alle posisjonene frem til brukeren som startet turen har mistet nettet.

5.1.3 Innloggingssystem

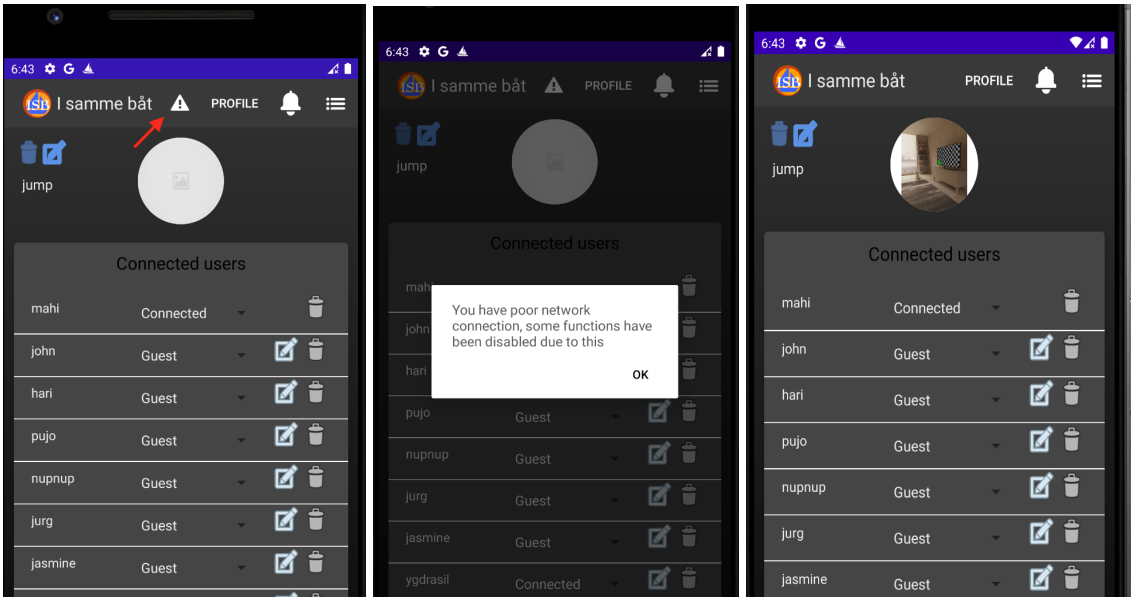
For å kunne logge inn brukere uten nett var vi nødt til å lagre passordet et sted på klienten, siden vi ikke kan sjekke data som ligger i databasen. Løsningen vi valgte var å lagre passordet lokalt i "SharedPreferences". Vi ville unngå å lagre passordet som ren tekst selv om det blir lagret i privat modus med en nøkkel som trengs for å få tilgang til det, og det er kun appen som har denne tilgangen. Dette har vi unngått ved å kryptere passordet og mail før de blir lagret lokalt ved bruk av en kryptonøkkel kalt "MasterKey" og "KeyGenParameterSpec". Det er nevnt i underkapittel 4.1.3 at i tilfelle brukeren ikke allerede var logget inn fra før, blir brukeren sendt til innloggingssiden for å logge inn manuelt. Dette tillater vi ikke uten dekning. Vi kunne ha sjekket mail og passord lokalt og tillatt manuell innlogging, men vi har bestemt oss for å unngå dette fordi vi heller vil validere hashet og saltet av passordet enn det krypterte passordet.

5.1.4 Design

Designet ble diskutert mye mellom teammedlemmene siden det er et veldig viktig aspekt i problemstillingen vår. For å forhindre at appen tar for stor lagringsplass lagrer vi ikke bilder lokalt med mindre de skal sendes til serveren. Teammedlemmene ble enige om et kompromiss for å ikke lagre altfor mye data lokalt og å designe appen slik at brukere legger minst mulig merke til forskjellen mellom offline og online modus. Det vi kom fram til var å vise et standardbilde (se figur 4.1) istedenfor å lagre alle bildene brukeren har i systemet, fordi dette fort kan ta veldig mye plass. Dette er grunnen til at vi har deaktivert å slette bilder på et logginnlegg uten dekning, da det er umulig å se hvilket bilde brukeren jobber med.

Brukeren kan fortsatt legge til bilder på et logginnlegg og slette disse om det er ønskelig. I tillegg kan brukeren slette og laste opp et nytt bilde på en båt siden det kun er ett bilde som ligger der. Vi er klar over at denne løsningen gir en mindre sømløs brukeropplevelse, men vi mener dette er fordelaktig for brukeren med tanke på lagringsplass.

Hvorvidt vi skal informere brukeren om at hen ikke har dekning har blitt diskutert mye. Måten dette informeres på ble implementert på flere måter før vi endte opp med den siste versjonen



Figur (5.1): skjermbilde som viser info-ikonet for å informere ved mangel på dekning

Figur (5.2): skjermbilde som viser meldingen brukeren får ved å trykke på info-ikonet

Figur (5.3): skjermbilde som viser appen uten varsel-ikonet

Første utkast av appen så ut som det blir vist i figurene 5.1 og 5.2 hvor vi viser en trekant med info-ikonet når brukeren mister nettet. Når brukeren trykker på ikonet blir en pop-up vist med nødvendig info som vist i figur 5.2. Dette alternativet ble vi enige om å fjerne fordi vi var redde for at løsningen kunne tolkes av brukeren som en hindring og står imot det problemstillingen vår handler om. Et scenario vi så for oss var at brukeren ville misforstå tilbakemeldingen og lukke applikasjonen fordi mobilen ikke har dekning lenger.

Et annet alternativ som ble diskutert var å gråe ut komponenter som ikke fungerer uten dekning, men dette ble heller ikke implementert grunnet det samme som ble nevnt i forrige avsnitt, nemlig å gjøre brukeren oppmerksom på at nettet er borte. Til slutt endte vi opp med å informere brukeren om at funksjoner ikke er tilgjengelige kun når brukeren prøver å benytte seg av dem. Med den nåværende versjonen vil den eneste indikasjonen på at enheten ikke har dekning være bildevisningen (se figur 4.1) såfremt at brukeren ikke prøver å bruke utilgjengelige funksjoner.

5.1.5 Brukertester (problemstilling)

Vi kjørte tre forskjellige testvarianter slik at vi fikk testet flest mulig funksjoner i appen slik at testene ikke tok for lang tid og for å forhindre at testbrukerne mistet fokus under testen. Testresultatene viste i stor grad at brukerne ikke la merke til noe særlig forskjell da vi slo av nettet. De fleste brukere som selv slo av nettet sa at de ikke ville lagt merke til noen forskjeller i applikasjonen om de ikke var klar over.

I løpet av testene fant vi et problem der en bruker skrudde av nettet før hen ga tillatelse til appen å spore posisjonen selv om dette burde kunne skje uavhengig av nettet. Dette problemet ble observert kun én gang i løpet av testing og utvikling. Et annet problem vi opplevde var at kartet ikke blir oppdatert da brukerne prøvde å zoome inn på kartet uten nett. Dette vil være umulig å løse da kartkomponenten ikke vil lagre data i lengre tid. Om brukeren har vært inne på kartet med nett, vil kun de delene av kartet som brukeren har utforsket være lastet inn når man ikke har nett.

5.2 Ingeniørfaglig diskusjon

I dette kapittelet skal vi diskutere hvorfor resultatene ble som de ble med tanke på de målene vi satte i visjonsdokumentet. Vi planla i førsteutkastet av visjonsdokumentet 41 ulike funksjoner som skulle implementeres i løpet av prosjektet. Dette var før vi hadde et fullstendig overblikk over prosjektet og visse aspekter ved appen ble endret i løpet av prosessen. Noen ble endret fordi vi ved utviklingen så at det var nødvendig med en endring, andre ble endret fordi produkteier kom med nye ønsker underveis i prosessen som ikke var klart i starten. Til slutt endte vi opp med 56 funksjoner. Vi var klar over fra starten av at noen av dem mest sannsynlig ikke ville bli implementert. Dette gjaldt funksjoner som å spore

forbruk av drivstoff på båten og å utvikle en tilhørende nettside for appen. Dette var mulige utvidelser om vi ble tidlig ferdig med alt det andre vi skulle. Tanken var at det var bedre å ha for mange funksjoner enn for få. I punktene under skal vi diskutere hvordan prosessen utfoldet seg, hvordan vi gikk fra 41 til 56 funksjoner og gå i dybden om hvorfor noen ble fullstendig implementert og andre ikke kom helt i mål.

5.2.1 Brukersystem

Ut fra oppgavebeskrivelsen var det tydelig at vi måtte ha et brukersystem i produktet. De to første funksjonene i punkt 5 i visjonsdokumentet beskriver registrering av bruker og innlogging, og det var også det første vi fikk på plass under arbeidet. Etter registrering har brukeren mulighet til å legge til et profilbilde, men det er ikke et krav og har heller ingen funksjon i resten av appen, siden vi endret hovedfokuset på appen fra brukere til båter.

Det er flere mulige utvidelser ved det sosiale aspektet til appen som vi ikke har implementert. Eksempler på dette er vennesystem, kommentarer og "likes". Produkteier var i starten ikke helt sikker på hvordan vennesystemet skulle utarte seg. Tanken var å enten utvikle et asynkront vennesystem med følgere eller et synkront vennesystem der man må godta en link mellom brukere. I løpet av prosjektet gikk vi bort fra begge deler og fokuserte mer på kobling mellom brukere og båter. All tilkobling mellom brukere gikk nå gjennom båter. Det var den viktigste koblingen vi trengte for funksjonene vi skulle implementere. Det er likevel en mulig utvidelse å implementere et vennesystem da det kan være nyttig for andre funksjoner vi ikke fikk implementert. Et eksempel på dette er funksjonen der brukere som befinner seg i nærheten av hverandre skal få varsel om det.

En mulig utvidelse relatert til vennesystemet vi har i visjonsdokumentet til dette var å kunne opprette grupper, gjerne mellom venner eller følgere fra vennesystemet som gjorde det enklere å dele turer med flere venner samtidig. Dette ble ikke en prioritet i prosjektet og ble ikke utviklet lenger enn idéstadiet. Meldingssystem var også noe vi planla å implementere for å bygge på brukerinteraksjoner, men ble nedprioritert etter ønske fra produkteier. Vi valgte å heller fokusere på et varselsystem som vi ikke tenkte på i planleggingsstadiet. Her vil man få varsler om man blir lagt til på en båt, en tur eller blir delt en tur med. Det er ikke en fullverdig erstatning av et vennesystem, men det tilfredsstiller noen av de samme behovene. Fra behovene vi har i visjonsdokumentet er for eksempel nr. 11 at man ønsker gode og enkle koblinger mellom brukere og båter, noe et varselsystem hjelper for.

En utvidet funksjon vi ikke kom helt i mål med var at brukere skulle akseptere å bli lagt til i en tur eller en båt. Den grunnleggende implementasjonen er på plass ved at en bruker får et varsel med en invitasjon og valget mellom å godta eller avslå. Om brukeren godtar går knappene vekk og brukeren er lagt til. Om brukeren avslår blir hen fjernet fra entiteten. Det vi ikke fikk implementert var at brukeren ikke skulle bli lagt til i entiteten før hen godtar invitasjonen. På grunn av dette får brukeren med en gang tilgang til entiteten og vil vises som tilknyttet båten eller turen uten å ha godtatt dette. Videre implementasjon ville vært å lagre om brukeren har akseptert i databasen og å ikke vise båten eller turen til brukeren før hen har godtatt koblingen.

I funksjonene fra visjonsdokumentet står det at man skal kunne se sin egen og andres profil. Vi startet med å ha profil som en av hovedsidene i appen, men mer og mer av brukerinformasjonen ble fjernet derfra. Produkteieren mente ikke at det var nødvendig og ville heller at båtinformasjonen skulle være mer sentral. Derfor ble profilsiden endret til en båtside og hele brukerprofilen satt til side i prosjektet. Det er derfor ikke implementert i det ferdige produktet og kan også vurderes som en mulig utvidelse. Til dags dato er det ingen visning av brukerinformasjon i appen utenom siden for endring av brukerinformasjon.

5.2.2 Båt- og havnesystem

Båtsystemet vi har implementert er ganske likt det beskrevet i visjonsdokumentet. I et større perspektiv ble appen mer sentrert rundt båter enn brukere, men bortsett fra det har vi samme grunnleggende implementasjon av systemet. Vi har implementert registrering, redigering og sletting av båt og å knytte brukere og havner til båt. Både registrering og redigering står i visjonsdokumentet, men sletting var en funksjon vi i senere tid fant ut var en god funksjon å ha. Det gir brukeren mer kontroll over dataen som blir lagret. Å kunne knytte brukere til båter ble heller ikke definert før vi var kommet godt i gang med prosjektet. I planleggingsfasen visste vi at det måtte være en kobling mellom brukere og båter, men planen var at det skulle fungere mer som et følgesystem hvor en bruker følger en båt. Etter å ha fått bedre oversikt over produktet måtte det endres. Produkteier la grunnlaget for roller vi har på båt, noe som førte til at vi satte funksjonen med å følge båter til side til fordel for dette rollesystemet.

Admin-brukere er gjerne båteier og det kan være flere som er deleiere av en båt. Vi var usikre på hvordan flere deleiere skulle implementeres, spesielt om hvorvidt en båteier burde få godkjenning fra de andre om hen skal endre noe på båten som er viktig for alle. Slik det er implementert nå kan alle adminbrukere slette båten eller fjerne andre adminbrukere fra båten uten at de andre har noe å si om det. Dette var etter ønske fra produkteier som mente det holdt for denne versjonen av appen.

En funksjon vi gikk helt vekk fra ganske tidlig i prosessen var å kunne måle og spore forbruk av olje, drivstoff og vann på båten. Å implementere dette ville vært svært tidkrevende dersom bruk av sensorer skulle bli involvert. Det var klart fra begynnelsen at dette var av veldig lav prioritet og skulle være en mulig tilleggsfunksjon om vi skulle få nok tid. Derfor er den nederst på lista over funksjoner i visjonsdokumentet.

Havner var i planleggingsfasen én enkeltfunksjon som ikke var definert mer enn at man skulle kunne manøvrere ruter ut fra havner på kartet. Senere i utviklingsfasen fikk vi en mer utdypende forklaring av hvordan det skulle implementeres fra produkteier. Derfor ble det planlagt og implementert senere i utviklingsprosessen. Denne ene funksjonen ble delt inn i flere mindre funksjoner som forklarer konseptet med havn på en mer utdypende måte. Disse står opplistet i kravdokumentasjonen vår, og i tabellen i kapittel 4.2 fra funksjon nr. 25 t.o.m. 29.

Vi har manuell oppretting, redigering og sletting av havn. Man kan redigere både navn og posisjonen til havn og man kan se havner underveis i turen. Havn blir automatisk opprettet både på avgangsstedet og på ankomststedet når man avslutter en tur om ikke det finnes en havn innenfor 50 meter. I utgangspunktet ønsket produkteier at denne lengden skulle være en variabel og skulle variere basert på om man var midt i en by med mange brygger eller om man var på en øde øy. Dette hang sammen med et ønske fra ham om at vi skulle hente data fra velihavn.no eller en annen lignende kilde for å ha en sentral liste med havner alle båter kunne bruke, men det ble for dårlig tid til å implementere det. Derfor er alle havnene vi nå har i appen direkte knyttet til én båt. En av hovedfunksjonene ved havner som produkteier ønsket var at turer skulle få navn på formatet “Fra [starthavn] til [slutthavn]”. Dette gir brukerne en mer oversiktlig og lett måte å finne fram til riktig tur i listen på hovedsiden.

En funksjon relatert til havn som vi bare fikk startet på var at vi skulle vise hvilken havn en båt ligger i. Det blir lagret hvilken havn en båt er i om den ikke er underveis i en tur, men det blir aldri vist til brukeren noe sted i appen. Om vi hadde hatt mer tid ville vi nok lagt det til som en ekstra bit med informasjon på båtsiden.

5.2.3 Tursystem

Tursystemet er en av kjernedelene i produktet og er derfor noe vi har fokusert mye på og ferdigstilt i mange aspekter. Hovedfunksjonaliteten med å kunne starte en tur, se ruten på kart både underveis og i ettertid og å kunne få visningen på et sjøkart var noe vi jobbet med tidlig i prosjektet og er alle på plass i det ferdige produktet. I planleggingsfasen hadde vi ikke definert alle begreper fullstendig så ordet logging er ikke helt tydelig definert i førsteutkastet av visjonsdokumentet. Flere av funksjonene som handler om å spore og lagre turen ble beskrevet som “logging av turen”.

Visning av turrute på kart ble gjort av kartkomponenten Mapbox. Mapbox gjorde det lett å plassere punkter og markører, og å utføre operasjoner på disse. I tillegg kunne kan man legge på nye lag med kart, som var nyttig da vi skulle få på plass sjøkart over norgeskysten. Som alternativ til kartkomponent kunne vi valgt Google Maps, men gikk bort fra det fordi det kostet penger å bruke.

Vi slet lenge med å få karttjenesten for sjøkart til å vises riktig i appen. Vi tenkte lenge at det var et problem som hadde med projeksjonen å gjøre. Dette tenkte vi fordi EPSG-koden vi bruker for å betegne projeksjonen er ulik den vi bruker i kartkomponenten. I tillegg dukket sjøkartet opp i appen, men ble vist på feil sted i forhold til verdenskartet som vises i bakgrunnen. Det viste seg etter mye undersøkelser, testing og litt hjelp fra produkteier at det var BBox attributtet som var feil. Her hadde vi prøvd å bruke statiske verdier siden det alltid var det som ble brukt i eksempler vi fant på nettet. Løsningen var å bruke “{bbox-epsg-3857}” som fungerer som en variabel som oppdaterer seg ut ifra hvilken del av kartet vi viser i kartkomponenten. Denne lille feilen ble det brukt veldig mye tid på å få fikset, som vi kunne brukt til noe annet.

Valget av Android som operativsystem var grunnet produkteiers oppfatning om at cross-plattform rammeverk ikke egner seg bra til GPS-funksjonaliteten vi trengte til dette

prosjektet, spesielt når appen skal kjøre i bakgrunnen. Det har funket bra og vi har hatt få problemer med posisjonssporing.

En mulig utvidelse vi hadde planlagt å implementere var at turer skulle kunne kategoriseres i større reiser. En tur ville da blitt mer som en etappe på en lengre reise. Denne funksjonen hadde ganske høy prioritet til å starte med, men da vi kom lengre ut i prosjektet ble det bestemt i samarbeid mellom produkteier og utviklerne at det skulle nedprioriteres for å heller prioritere havnekonseptet. Derfor har vi en ubrukt entitet i databasen som heter "journey", som ville vært reise-tabellen. Det ville vært en av de neste funksjonene vi ville implementert om vi hadde hatt mer tid.

En siste funksjon som ble nedprioritert og derfor ikke fullført var å kunne bytte mellom ulike typer kart. Hovedideen med appen var å bruke sjøkart for å vise ruta til turen, men det kom også inn ønske fra produkteier om å kunne bytte kart til for eksempel satellittbilder eller andre naturkart som vil vise egenskaper sjøkartet ikke viser. Vi startet så vidt med å se etter gode kilder til andre karttjenester, men fant ingen gode på kort tid. Produkteier mente også at dette var av veldig lav prioritet og derfor ble det ikke jobbet noe mer med det.

5.2.4 Logginnleggsystem

Alle planlagte funksjoner relatert til logginnlegg er implementert. Hvem som har rettighet til hva når det gjelder skriving, sletting og redigering av logginnlegg er beskrevet i figur 0.1 og 0.2. Etterhvert i utviklingen ble det fremsatt et ønske fra produkteier om å ha muligheten til å skrive logginnlegg som ikke tilhører en tur, men som kun er knyttet til båten. Et eksempel på en situasjon hvor dette kan være nyttig er hvis en båteier er innom for å gjøre vedlikehold av båten og vil loggføre dette.

Logginnlegg vises på tilhørende posisjon på kartet hvor vi ser ruta til turen, og kan også vises som en liste i kronologisk rekkefølge. Alle logginnlegg en bruker har tilgang til vises også på hjemmesiden i en egen boks markert med logginnlegg. Her ligger de systematisert under først båt og deretter under turen de ble skrevet på om det var underveis på en tur. Turene og logginnleggene blir sortert kronologisk, med de nyeste først. Denne visningen på hovedsiden ble implementert etter ønske fra produkteieren, og skal minne om formatet på en analog loggbok.

Vi hadde litt problemer med lagring av bilder i databasen underveis i prosjektet. Da vi prøvde å sende med bilder som ble tatt på ekte enheter fikk vi statuskode “413 Payload Too Large” tilbake fra serveren, som betyr at vi sendte for mye data. For å løse dette gjorde vi to ting. For det første sørget vi for at bare ett bilde ble sendt med hver forespørsel til serveren istedenfor å sende alle bildene til et logginnlegg samtidig. For det andre komprimerer vi bildene før de sendes, noe som gjør bildekvaliteten litt dårligere. For å slippe å bruke alt for mye tid på dette, valgte vi å gå for denne løsningen.

5.2.5 Brukertester (design og struktur)

Som tidligere nevnt kjørte vi 13 tester med tre forskjellige varianter for å teste funksjonaliteter i appen mest mulig. Vi fikk mange tilbakemeldinger på design, hvor de aller fleste i hovedsak var positive. Mange slet med å komme seg til hovedsiden hvor alle logginnleggene er og hvor man starter en ny tur. De skjønte ikke at man måtte trykke på logoen eller navnet på appen for å komme seg dit. For å gjøre det lettere for brukerne la vi til et husikon som skulle representere hovedsiden. Nå kan man trykke på både logoen, appnavnet og husikonet for å komme dit.

Et annet problem flere hadde med navigasjonen var at de syntes det var vanskelig å komme seg tilbake uten en dedikert tilbakeknapp på siden til tross for at Android-telefoner har en egen tilbakeknapp innebygd i designet sitt. Det er litt vanskelig å vite om dette er en reell feil ved navigeringen da flere av testbrukerne ikke bruker Android, men iPhone til vanlig. Appen er bygd for Android og Android-brukere er mer vant til å bruke tilbakeknappen. Om vi hadde hatt mer tid til å prioritere småting som dette kunne vi lagt til en tilbakeknapp flere steder for å gjøre appen mer tilgjengelig for alle.

Flere slet med å finne hvor man kunne søke etter logginnlegg i nærheten og mente at forstørrelsesglassikonet ikke var beskrivende nok til den funksjonen. Vi fant ingen god løsning på dette problemet og endte opp med å beholde forstørrelsesglasset. En testbruker foreslo å endre posisjonen til ikonet fra detaljer-boksen til å stå øverst på kartet. Det kan hende det hadde hjulpet for å gjøre hensikten med knappen mer intuitiv, men vi hadde ikke nok tid til å kjøre nye brukertester for å sjekke om det ville funket eller ikke. Lignende tilbakemeldinger fikk vi angående knappen for å legge til et nytt logginnlegg som noen også

mente ikke var helt beskrivende, mens andre tenkte det måtte være det knappen var til. Vi fant heller ingen god løsning på dette. Vi kunne kanskje endret knappen fra ikon til ren tekst, men vi følte det ville skadet designet og strukturen ved å ha enda en knapp med tekst som midtpunkt i pågående tur-skjermen.

Et par av testbrukerne forstod heller ikke helt hva "Mine båter"-knappen i verktøylinja betydde. Dette kan være forårsaket av at mange av testbrukerne ikke hadde båt selv og ikke hadde helt samme tankegang som en reell bruker ville hatt. Vi valgte likevel å bytte ut teksten til et ikon av en båt for å gjøre designet på verktøylinja mer universelt og få det til å passe bedre inn sammen med husikonet og bjelleikonet.

Som tidligere nevnt var det flere testbrukere som med vilje prøvde å finne mangler ved systemet ved å skrive inn tekst i innskrivningsfelter som ikke var gyldig. Da oppdaget vi at noen felter tillot lengre tekst enn det attributtene i databasen tillot. Etter denne feilen ble oppdaget gikk vi gjennom alle felter for innfylling og lagde en grense for antall karakterer som stemmer overens med begrensningene vi har satt i databasen. Testbrukerne reagerte også på at vi ikke hadde stor bokstav som en standard i innskrivingsfelter. Etter testene endret vi dette i alle felter unntatt der mailadresse skal skrives for da vil man ikke nødvendigvis skrive stor bokstav først.

Noen testbrukere reagerte på at appen ga få tilbakemeldinger, for eksempel ved registrering av ny bruker. For å bedre dette gikk vi gjennom alle aktivitetene og ga brukeren tilbakemelding om feil når klienten får en feilmelding fra serveren og også en bekreftelse når ting går som de skal.

Flere var også ikke helt sikker på om turen faktisk hadde startet da de trykket på "Start tur", siden appen ikke ga noen beskjed om det. Vi skulle derfor gjerne endret hvordan designet er når man starter en tur. Tidligere hadde vi et stort hjelpevindu som ga brukeren mulighet til å fylle inn tittel og beskrivelse med mer ved starten av turen. Da var det mer tydelig når en tur faktisk ble startet. Produkteier ønsket derimot at vi skulle fjerne dette for å gjøre det raskere for en bruker å starte en tur. De som ønsker å fylle inn informasjon kan heller gjøre dette i redigeringsvinduet etter at turen er startet. Denne endringen skjedde ganske sent i prosessen, så vi fikk ikke vurdert nøye nok hvordan dette ville påvirke brukeropplevelsen før vi hadde brukertester. For å gjøre det mer tydelig at turen er i gang burde vi kanskje lagt til en

visningsboks som formidler dette. Kanskje ved hjelp av en klokke som viser hvor lenge turen har pågått og et rødt ikon som viser at man er i "opptaksmodus". Vi er ikke helt sikre på hvordan vi skulle løst det, men det kunne vært mulige løsninger vi kunne prøvd oss på.

Det var alt i alt veldig hjelpsomt å holde brukertester. Ikke bare for å teste problemstillingen og undersøke om brukeren ville merke at vi slo av nettet, men også for å prøve ut designet og strukturen vår og se om alt var intuitivt og lett å forstå. Det var flere rettelser som ble gjort på grunn av testene og appen ble mye bedre etter det. Dessverre var det også en del ting vi ikke fikk fikset, primært på grunn av liten tid når det var ganske store endringer som måtte til for å fikse problemene.

5.3 Administrativ diskusjon

Ved å bruke Scrum som utviklingsmetodikk satte vi underveis i prosjektet tydelige mål for når ulike funksjonaliteter skulle være implementert. Vi klarte å oppnå de fleste målene vi hadde satt for hver sprint, men det var ofte enkelte funksjoner som ikke var helt ferdige i løpet av sprinten de var satt opp på. For eksempel rakk vi ikke å begynne med posisjonssporing og sjøkart i løpet av sprint 1, selv om dette var satt opp i sprintbackloggen. Dette var delvis grunnet at vi heller ville skrive opp flere funksjoner enn det vi kom til å rekke enn færre. Det var imidlertid noen arbeidsoppgaver som ble utsatt og nedprioritert til tross for at de sto på sprintbackloggen fra starten. Den største av disse er testing av kode, som vi ikke kom skikkelig i gang med før sprint 5, til tross for at vi hadde skrevet opp testing som en arbeidsoppgave under alle funksjoner siden sprint 1.

En ting vi burde ha planlagt bedre tidligere i prosjektet er arbeid med problemstillingen. Det var først ganske sent i prosjektet at det ble klart for oss hvor omfattende dette arbeidet skulle bli, og dermed endte vi opp med å jobbe veldig intensivt med å få applikasjonen til å fungere uten nett mens andre funksjoner ble satt på vent. Vi hadde ikke sett for oss at dette aspektet ved applikasjonen skulle bli så tidkrevende. Noen funksjoner har derfor endt opp med å ikke bli fullstendig implementert, slik som bekreftelse om at man er tilknyttet en båt eller tur. Muligheten for å arkivere flere turer i samme reise var en funksjon produkteier ønsket og som vi selv ga middels prioritet i produktbackloggen. Denne funksjonen valgte vi å ikke begynne på, delvis grunnet arbeidet med problemstillingen.

Ved å bruke Scrum fikk vi en strukturert måte å arbeide på, hvor vi ved hjelp av sprintbackloggen alltid hadde klart for oss hva som måtte gjøres. I sprintbackloggen ble alle funksjoner oppdelt i ulike arbeidsoppgaver, og den ble oppdatert på slutten av hver dag med en ny estimering av hvor mange timer man hadde igjen på hver oppgave. På slutten av hver sprint hadde vi sprint-demo og sprint-review med produkteier og veileder. Ved å sette overordnede mål for hver sprint, slik som beskrevet i avsnittet over, ble vi motivert til å jobbe for å bli ferdig med funksjonene før sprint-demo.

I tillegg hadde teammedlemmene sprint-retrospektiv i slutten av hver sprint. Her diskuterte vi hva som hadde gått bra i løpet av sprinten, og hva som kunne ha gått bedre. Etter å ha diskutert og blitt enige om disse punktene, kom vi fram til ting vi ville fortsette å gjøre i neste sprint. Vi diskuterte også eventuelle ting vi skulle prøve å gjøre annerledes i neste sprint, og vurderte hvordan det hadde gått med de tilsvarende tingene vi hadde kommet fram til fra forrige sprint. Ved å gjennomføre sprint-retrospektiver ble vi oppmerksom på ting som kunne forbedres som ikke hadde med selve utviklingen av produktet å gjøre, men med selve teamarbeidet. Et godt teamarbeid er en forutsetning for et godt produkt.

Som resultat av sprint-retrospektivene fikk vi styrket samarbeidet og arbeidsmetodene våre. Noen nevneverdige punkter vi forbedret var bestemmelser om å ha faste og flere pauser, committe oftere for å unngå merge-konflikter og opprettelsen av et eget dokument med oversikt over mindre arbeidsoppgaver vi måtte jobbe med. Sistnevnte ble vår egen versjon av et issueboard. Her satte vi opp problemer, feil vi støtte på og mindre deler av funksjoner vi måtte implementere inndelt i ulik prioritetsrangering. Dette hjalp oss med å holde oversikt over hva som måtte gjøres på kort sikt og hva de andre teammedlemmene jobbet med for øyeblikket siden vi markerte punktene når vi startet på dem.

Prosjektets oppdeling i sprinter med 7 til 11 arbeidsdager var stort sett hensiktsmessig, slik at alle dagene utgjorde en uavbrutt sekvens med dager (med unntak av helger). Det var imidlertid ett tilfelle hvor vi kunne ha gjort mer for å få en kontinuerlig sprint. Sprint 4 besto av 11 arbeidsdager, men varte helt fra 23. mars til 15. april. Den ble dermed ganske oppstykket, med flere store mellomrom mellom dager vi jobbet. Dette var delvis grunnet ting vi ikke hadde så mye kontroll over. I løpet av perioden hadde både en eksamen i et annet fag og påskeferie en uke etter dette. Vi kunne likevel ha planlagt bedre for å ikke la denne

sprinten gå over en så stor periode, siden det er lett å miste fokus og oversikt når man har store pauser mens man jobber med samme funksjon.

5.4 Helhetlig system

For bruk av applikasjonen kreves det at brukeren benytter seg av en Android-mobil. For å kunne benytte seg av posisjonssporing, som er en viktig del av kjernefunksjonaliteten, er brukere nødt til å gi appen tilgang til deres posisjon. I tillegg må brukerne gi appen tilgang til kamera eller galleri om de vil legge inn bilder. Ellers er alle øvrige funksjoner tilgjengelige for brukerne dersom de har registrert konto i systemet. I motsetning til en vanlig båtlogg der brukerne skriver all info om turer og båten på et ark, kan de gjøre det i appen slik at de lett kan dele turer og logginnlegg med andre bekjente og familiemedlemmer. Dette gjør appen til både et sosialt medium og et arkiv brukerne kan benytte til å enkelt se gjennom minner og info de har skrevet om båten. Applikasjonen tar høyde for personvern da ingen info blir lagret om brukeren bortsett fra mail og navn, i tillegg til at brukeren har full kontroll over sin egen data og kan når som helst fjerne data hen ikke ønsker å ha liggende i systemet.

Til tross for at applikasjonen har sjøkart er den ikke ment til å brukes til navigasjon. Vi kan ikke garantere at posisjonssporing alltid fungerer helt perfekt, og det kan oppstå tilfeller hvor posisjonsmarkøren for båten vises på feil sted. Vi utviklere har heller ingen planer om å kontinuerlig følge med om kartet er oppdatert og stemmer med virkeligheten. Derfor kan vi ikke garantere at alle detaljer på kartet til en hver tid er riktig. Dette informerer vi brukeren om i applikasjonens hjelpeside. Ved å informere om dette blir brukeren inneforstått med at kartet brukes på eget ansvar.

Det er usikkert hva som vil skje med appen etter at prosjektet er ferdig, men hvis det blir lansert på markedet kan det potensielt tilby en kostnadseffektiv måte for brukeren å benytte seg av sjøkart, spesielt sammenliknet med fysiske kart.

Kapittel 6: Konklusjon og videre arbeid

6.1 Konklusjon

Problemstillingen vi presenterte i innledningen av rapporten er som følger:

> Hvordan kan vi designe og implementere en nettverksbasert mobilapplikasjon som opprettholder en god brukeropplevelse selv ved redusert eller mangel på nettverksforbindelse?

For å kunne implementere en applikasjon som er sømløs og opprettholder en god brukeropplevelse uten nett var vi nødt til å ha på plass en app som fungerer bra med nett. Dette var en tidkrevende prosess hvor vi prøvde å få på plass så mange funksjonaliteter som mulig før vi prøvde å gjøre dem fungerende uten nett. Etter rådgivning fra veileder ble problemstillingen utvidet ganske sent i prosessen fra å gjelde kun posisjonssporing til å omhandle all hovedfunksjonalitet. Omfanget av problemstillingen ble mye større enn vi først forventet, noe som førte til at tiden ikke helt var på vår side. Prioriteringene endret seg veldig etter at vi hadde klar problemstillingen som førte til at andre funksjoner ble satt på vent.

En av de store utfordringene med problemstillingen var å lagre all nødvendig data på brukerens enhet, og å bruke denne på riktig sted når brukeren ikke har nett. I tillegg til dataen som blir lagret i databasen før forbindelsen blir brutt, må også data som brukeren legger til etter å ha mistet nett lagres på enheten helt til brukeren får nett igjen. Alt dette må skje på en slik måte at brukeren merker minst mulig forskjell fra om enheten har dekning eller ikke.

Det var viktig for oss å ha en balanse mellom å la grensesnittet være uavhengig av nettverksforbindelse, men samtidig ikke lagre alt for mye data på brukerens enhet. Blant annet lagrer vi ikke bilder og forhindrer dobbeltlagring av endringer som skal sendes til databasen. Dette økte kompleksiteten til koden, noe som gjorde det vanskeligere å jobbe med.

I hver klasse som hadde kontakt med serveren i applikasjonen måtte vi ha en egen metode som håndterte seg av forespørsler som ikke ble sendt pga. mangel på nett, og hvor relevant data enten ble hentet fra eller skrevet til fil.

Problemstillingen omfatter opprettholdelse av brukeropplevelse både ved redusert nettverkstilkobling, og ved mangel på det. Vi har ikke hatt mulighet til å teste hvordan applikasjonen fungerer når nettverkstilkoblingen kun er redusert, men ikke borte, for eksempel med EDGE-dekning. Det å teste dette er vanskelig av natur, da man ikke kan konfigurere mobilen til å ha dårlig dekning, men må være et sted der mobilnettet ikke er tilstrekkelig.

Ut ifra testresultater og tilbakemeldinger kan vi si at hovedfunksjonaliteten i appen fungerer uavhengig av nettverksforbindelse ved bruk av lokal lagring. En mulig negativ konsekvens ved måten vi har løst problemstillingen på er at oppstartstiden til applikasjonen øker kraftig når vi må hente mye informasjon for å være forberedt på mangelfull dekning. I tillegg mener teammedlemmene at appen ikke er feilfri og fremdeles trenger videreutvikling og mer testing (særlig ute på sjøen der applikasjonen kommer til å bli brukt), før den settes i drift.

Vi vil anbefale utviklere som skal lage en lignende applikasjon til å planlegge at funksjoner skal fungere uten dekning helt fra starten. Dette kan gjøre arbeidet lettere og gir oversikt over hele bildet av systemet og hvordan hver funksjonalitet bør implementeres slik at det fungerer like effektivt uavhengig av nettverksforbindelse.

6.2 Videre arbeid

Ved videreføring av prosjektet må det legges en plan for hvilke av de planlagte funksjonene som skal implementeres. Flere av funksjonene fra visjonsdokumentet er delvis fullstendige og noen er ikke implementert i det hele tatt. Det er en vurdering som må gjennomgås for å finne ut om appen trenger et eget system for å sammenkoble brukere synkront eller asynkront, eller om det holder med koblingene mellom båt og bruker som er implementert til nå. Uansett burde profilvisning implementeres hvor man kan se sin egen og andres profil. Her kunne det også ha blitt vist turene brukerne har vært på sammen.

Konseptet rundt reiser burde implementeres og muligens ha sin egen seksjon i appen. Sortering og organisering av turer kan gjøre appen mer strukturert og oversiktlig. Det vil også styrke appens egenskap til å fungere som et arkiv.

Av de andre funksjonene vi ikke har fått implementert mener vi at disse har størst prioritet til å videreutvikles:

- Brukere får varsel om bekjente brukere er i nærheten
- Brukeren får mulighet til å verifisere mail før kontoen blir aktivert
- At brukere må akseptere invitasjoner til båt og tur før de får tilgang
- Bytte av kartvisning
- Logge inn med kontoer fra andre sosiale medier

Kapittel 7: Referanser

Bartnes, Maria (2019) *HTTPS*. Tilgjengelig fra: https://snl.no/HTTPS (Hentet: 14. mai 2021)

Beck, K. *et al.* (2001) *Manifesto for Agile Software Development*. Tilgjengelig fra: https://agilemanifesto.org (Hentet: 13. mai 2021)

Fielding, R.T. (2000) *Architectural Styles and the Design of Network-based Software Architectures*. PhD. University of California, Irvine. Tilgjengelig fra: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm (Hentet: 13. mai 2021)

Forsell, B. og Kjerstad, N. (2021) *GPS*. Tilgjengelig fra: https://snl.no/GPS (Hentet: 14. mai 2021)

Internet Engineering Task Force (IETF) (1999) *RFC 2616*. Tilgjengelig fra: https://datatracker.ietf.org/doc/html/rfc2616#section-10 (Hentet: 13. mai 2021)

Internet Engineering Task Force (IETF) (2014) *RFC 7231*. Tilgjengelig fra: https://datatracker.ietf.org/doc/html/rfc7231#section-4.3 (Hentet: 13. mai 2021)

Personopplysningsloven (2016) *Lov om behandling av personopplysninger*. Tilgjengelig fra: https://lovdata.no/dokument/NL/lov/2018-06-15-38/KAPITTEL_gdpr-2#gdpr/a5 (Hentet: 13. mai 2021)

EncryptedSharedPreferences (u.å.) Tilgjengelig fra: https://developer.android.com/reference/kotlin/androidx/security/crypto/EncryptedSharedPreferences.html (Hentet: 16. mai 2021)

H2 Database Engine (2019) Tilgjengelig fra: https://h2database.com/html/main.html (Hentet: 14. mai 2021)

KeyGenParameterSpec (u.å.) Tilgjengelig fra: https://developer.android.com/reference/android/security/keystore/KeyGenParameterSpec.html (Hentet: 16. mai 2021)

MasterKey (u.å.) Tilgjengelig fra: https://developer.android.com/reference/androidx/security/crypto/MasterKey (Hentet: 16. mai 2021)

Password Hashing (u.å.) Tilgjengelig fra: https://docs.oracle.com/cd/E26180_01/Platform.94/ATGPersProgGuide/html/s0506passwordhashing01.html (Hentet: 14. mai 2021)

Quarkus - Datasources (u.å.) Tilgjengelig fra: https://quarkus.io/guides/datasource#h2 (Hentet: 14. mai 2021)

Quarkus - Testing your application (u.å.) Tilgjengelig fra: https://quarkus.io/guides/getting-started-testing (Hentet: 14. mai 2021)

Save key-value data (u.å.) Tilgjengelig fra: https://developer.android.com/training/data-storage/shared-preferences (Hentet: 16. mai 2021)

SLF4J user manual (2019) Tilgjengelig fra: http://slf4j.org/manual.html (Hentet: 14. mai 2021)

What is a relational database (2021) Tilgjengelig fra: https://www.oracle.com/database/what-is-a-relational-database/ (Hentet: 13. mai 2021)

Kapittel 8: Vedlegg

8.1 Oppgaveteksten

Arbeidstittel:	I samme båt
Hensikten med oppgaven: Utvikle en app for Android og / eller iOS og tilhørende serverkomponenter.	
Kort beskrivelse av oppgaveforslag: Loggboka er en viktig komponent i et båthold. Den dekker turbeskrivelser, praktisk informasjon med mer, og kan fungere som "hyttebok". Det finnes i dag ingen gode digitale løsninger som dekker alle disse aspektene på en god måte, og ingen som ivaretar det sosiale aspektet som er til stede i båtlivet. Oppgaven er å utvikle en app som har som kjernefunksjonalitet å logge en båtreise med posisjonsspor, og ta tekst og bilder under veis som knyttes til spesielle hendelser. Innholdet skal kunne deles med medreisende og andre.	
Oppgaven passer for (kryss av de(t) som passer og skriv evt. en kommentar til oss):	- Bacheloroppgave
Kan oppgavestiller stille arbeidsplass med nødvendig utstyr og programvare:	Ja
Hvis ikke, hva kreves av maskin og programvare:	Bør kunne teste app på egen telefon
Oppgaven passer best for, antall studenter:	- 3 - 4
Opplysninger om oppgavestiller	
Er du fra bedrift/virksomhet eller er du student med en egendefinert/selvlaget oppgave?	- Bedrift/virksomhet
Navn på bedrift/organisasjon/student:	Kantega AS
Addresse	Bassengbakken 4
Postnummer	7042
Poststed	Trondheim
Navn på kontaktperson/veileder:	Håvard Wigtil
Telefon:	93846468
Epost:	haavard.wigtil@kantega.no

Utfyllende kommentarer til hva oppgaven gjelder:
Oppgaven er å utvikle kjernen til en god digital loggbok for båt. Det er naturlig å starte med det en mobiltelefon er god på, som å løpende registrere posisjon, knytte tekst og bilder til posisjon, og å måle og vise fart og seilt distanse. Visning av posisjon bør være integrert med sjøkart.

Å dele båt er en moderne og kostnadseffektiv måte å eie båt på, så det må fra starten legges til rette for at en båt kan ha flere primære brukere og tilknyttet mannskap. Det er også naturlig å tenke at posisjon eller logg skal kunne deles med venner på andre båter.

Opptak og rapportering av posisjon må være robust også når applikasjonen er i bakgrunnen eller dekning er dårlig, og det er viktig å balansere batteribruk og god sporing. Dette sammen tilsier at dette må være en native app, heller enn implementert med React Native o.l. App utvikler enten med Kotlin / Java eller Swift.

Server-komponenten skal utvikles i Kotlin eller Java. Driftsform vil bli bestemt ved oppstart, men det er naturlig med

en form for skybasert drift.

Det er mulig å tenke seg mange utvidelser, f.eks. nettside for bedre visning av turer, sporing av forbruk (diesel, olje, vann), å utvikle den sosiale dimensjonen med relasjoner mellom båter og mannskap, og så videre. Hvis det er tid til utvidelser ut over kjernekonseptet vil disse bli håndtert etter smidig-prinsipper i samarbeid mellom utviklere og produkteier.

Systemdokumentasjon

I samme båt

Anvendt Informasjonsteknologi, IDI, NTNU

TDAT3001 Bacheloroppgave vår 2021

Revisjonshistorie

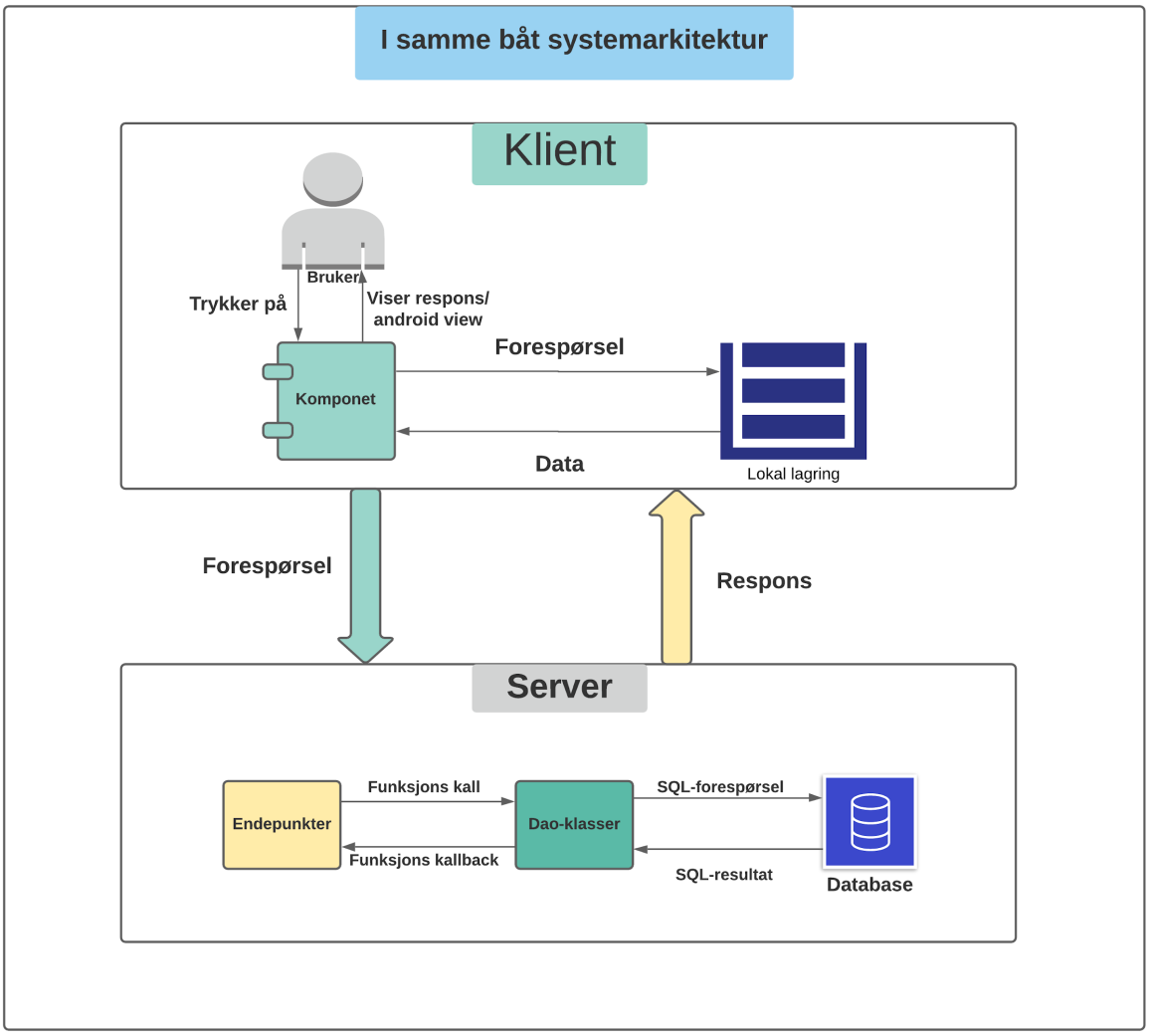
Dato	Versjon	Beskrivelse	Forfatter
04/05/21	1.0	Mesteparten av punktene er på plass	Mahmoud Ibrahim, Eirik Plahte og Christian Riksvold
17/05/21	2.0	Renskriving av dokumentet	Mahmoud Ibrahim, Eirik Plahte og Christian Riksvold

Innholdsfortegnelse

1. Introduksjon

Dette dokumentet er skrevet i forbindelse med prosjektet “I samme båt”, som er laget for å oppfylle kravene til emnet TDAT3001 Bacheloroppgave dataingeniør (2021 vår) i studieprogrammet Bachelor i ingeniørfag, data.

Dokumentet inneholder systemdokumentasjon til systemet “I samme båt”, blant annet systemarkitektur, klassediagram, databasemodellering og alt under innholdsfortegnelse. I tillegg blir dokumentet brukt for kommunikasjon mellom utviklere og både oppdragsgiver og veileder. Dokumentet skal også oppsummere alle valgene som ble tatt til å bygge opp systemet og begrunne hvorfor teammedlemmer har valgt dem.



Figur 2.1 systemarkitektur som viser dataflyt mellom klient og server

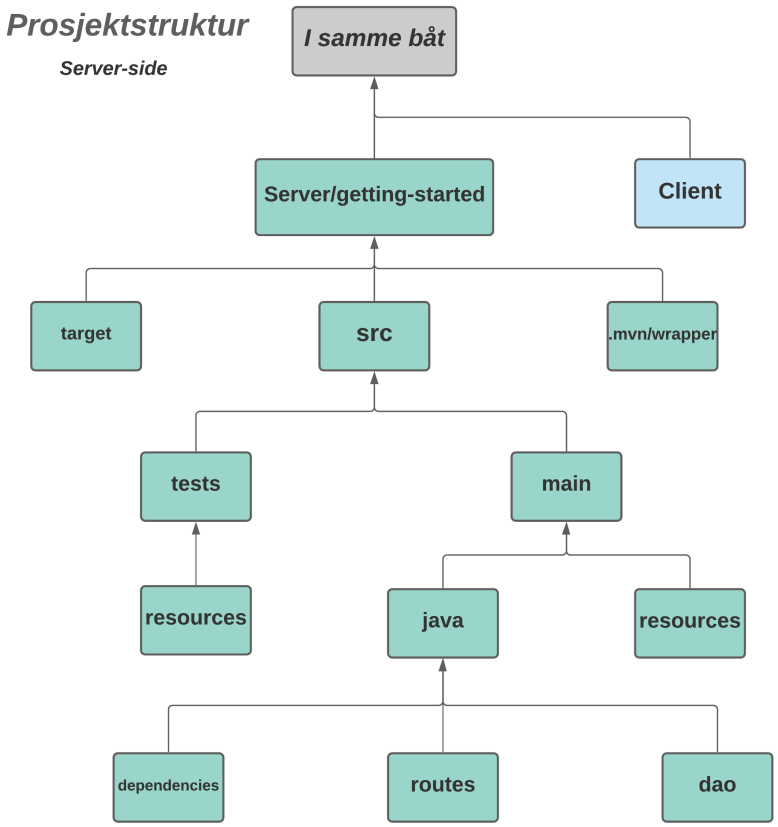
Figuren (2.1) viser helheten i applikasjonen og hvordan data sendes mellom klient og server. Figuren gir også detaljert oversikt over server og klient hver for seg og hvordan de ulike komponentene kommuniserer sammen og dataflyt mellom dem. Ser man på det generelle bilde i figuren, så ser man at kommunikasjonen skjer slik at klienten sender forespørsel til serveren og får responsen tilbake.

På klientsiden ser vi at når brukeren trykker på Android-komponenten sendes en forespørsel til enten serveren eller lokal lagring som er sharedpreferences i vårt tilfelle avhengig av internett dekning på mobilen. Dersom det ikke er noe dekning blir alle forespørsler skrevet til

og hentet fra lokal lagringen frem til mobilen kobles til nettet på nytt . Hvis mobilen er koblet til internett sendes alle forespørslene til serveren. Alle forespørslene blir sendt med et token uavhengig om mobilen er koblet til internett eller ikke unntatt forespørselen som sendes til å registrere og logge inn endepunkter. Før klienten sender forespørsel til server blir tokenet hentet fra lokal lagring og sent med forespørsel. Etter at klienten har fått respons fra serveren reagerer android komponenter basert på responsen og gir tilbakemelding til brukeren ut i fra responsen. I tillegg blir det nye tokenet som klienten fikk i responsen sett i lokal lagring og klar for å bli hentet for neste forespørsel Tilbakemelding kan være bekreftelse, feilmelding eller vise data avhengig av pakken og http forespørsel som blir sendt til serveren

I figuren over ser vi også på hvordan data flyter i server-side. Helt til venstre har vi endepunktene våre som tar i mot forespørsler fra klienten via “Http wrapper”. Et eksempel på det kan være å registrere en tur. Endepunktet tar det forespørselen og sjekker om brukeren er autorisert eller ikke. Dersom ikke, sender den *respons med status 401* til klienten, eller status 400 “bad request” dersom brukeren ikke har en aktiv båt . Men hvis brukeren er autorisert og har en aktiv båt kaller endepunktet funksjonen i dao klassen til å kjøre SQL-skriptet mot databasen. Derfra sender databasen resultater til dao som videresender callbacks til endepunktet for å sende responsen til klienten. Når klienten har fått responsen settes det nye tokenet i sharedpreferences til å bli brukt i neste forespørsel. Til slutt vises data til brukeren via android komponenter avhengig av responsen og status som ble sendt fra server

3. Prosjektstruktur



Figur 3.1 prosjektstruktur på serversiden

I figuren (3.1) viser vi oversikt over alle viktige mappe vi har i server-side. Figuren viser også at server-klient arkitektur blir implementert og brukt, men vi har valgt å ikke vise hele prosjektstruktur til både server og klient i én figur for å gjøre det lettere for leseren å ha god oversikt over hver del og å gjøre det lettere å forstå.

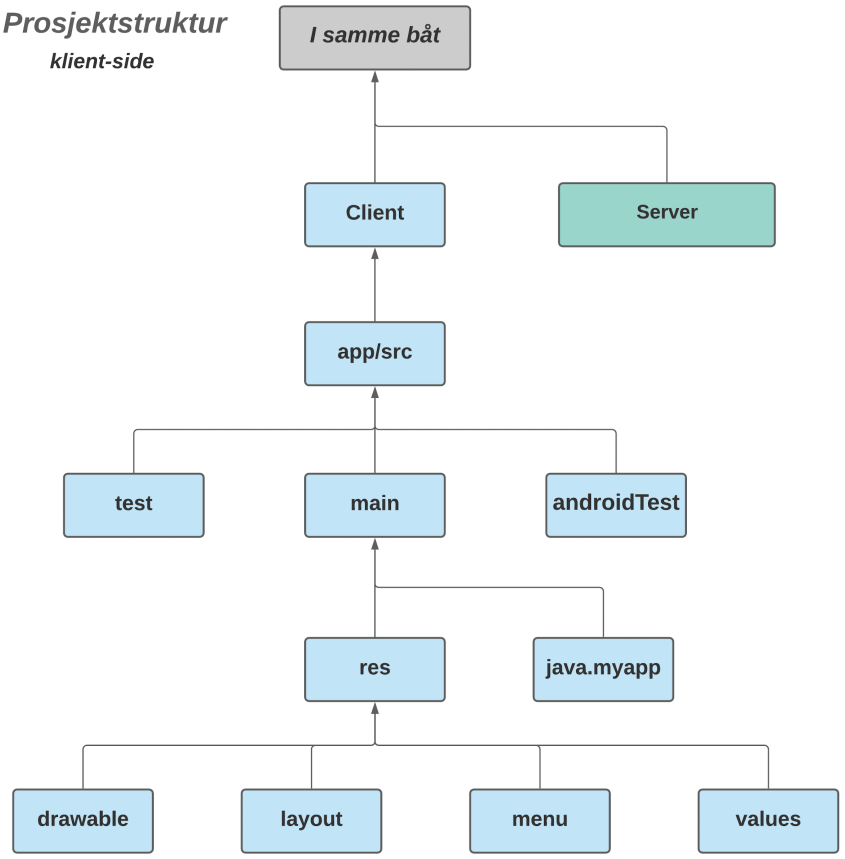
Under *server/getting-started* mappe har vi tre hoved mapper i tillegg til *pom.xml* som inneholder alle konfigurasjoner for å bygge et maven prosjekt i serveren. De tre forskjellige mappene er:

1. **mvn/wrapper:** den inneholder maven wrapper. Maven wrapper brukes til å kjøre et spesifikt wrapper-prosjekt i stedet for å installere mange forskjellige versjoner av maven.
2. ***target:*** den inneholder *index.html-fil* som viser dekning av server-testing, og en jar-fil til som brukes til å deploye quarkus til google cloud. Les mer om dette her.

- **Index.html:** fil blir generert via jacoco som er et programvare berging som vi har brukt til å måle hvor mange linjer i koden vår blir utført under automatiske testene vi har i serveren.
- **"*getting-started-1.0.0-SNAPSHOT-runner.jar*"**: den blir laget til ved å bygge prosjektet. Denne filen blir brukt til å deploye serveren (quarkus prosjekt) til Google App Engine Standard .

3. ***src*** er en stor mappe i servers-side som inneholder alle klassene og testene vi har skrevet for å bygge systemet "i samme båt" . Den mappa inneholder to mapper som følgende er

- ***tests*** som inneholder alle alle testfiler til endepunkter vi har i systemet. I tillegg til det har *tsets* en undermappe som er
    - ***resources***: som undermappe, og den inneholder alle nødvendige filer til å få testene til å kjøre. Blant annet *import.sql* som inneholder sql-scriptene med testdata som trenges til å kjøre testene. application.properties hvor vi defenerer alle kofigrasjons variablene for testene våre som f.eks. hvilken type database vi bruker, brukernavn og passord til databasen, urlen som forteller kompilator om hvor ligger scriptet som skal kjøres.
- ***main***: største mappen i systemet vårt i server-side og som er mest viktig. Den mappa har to følgende undermapper
    - ***java***: under den mappa har vi *dao-mappe* for alle dao-klassene vårt, *routes-mappe* for alle endepunkter-klassene i systemet vårt og *dependencies* som har alle jar-filene til bibliotekene vi brukes i serveren f.eks jacoco, mysql connector, slf4d.
    - ***resources***
        - ***META-INF***: som inneholder index.xml-fil som blir generert når serveren er depolyet til google cloud for å se på data som vi får tilbake fra endepunkter i google cloud appen, men det viser ingenting bortsett fra en tilbakemelding på at server har blitt deployert. Grunnen til dette er at vi ikke har noen endepunkter som er direkte til google cloud, men i stedet deployer hele serveren i prosjektet vårt til google
        - ***application.properties*** hvor vi defenerer alle kofigrasjons variablene for serveren



Figur 3.2 prosjektstruktur på klientsiden

Figuren over viser strukturen for klient-delene som har undermappa app/src. Den mappa fungerer som root mappa i klientside og har alle gradle innstillinger og tre undermapper som det vises i figuren

1. main: som er det største mappa i klientside og har to undermapper. Den første er java.myapp som har alle java-klassene til aktiviteter vi har i "i samme båt" som f.eks. Trip, Home, Login...osv. Den andre mappa er res som inneholder følgende fire undermapper

- drawable: har alle ikoner og bilder blir brukt i applikasjonen vår.
- layout: har alle xml-filene som er laget i applikasjonen vår, det vil si alle skjermene, knappene, android komponenter og layouts brukeren ser i applikasjonen
- menu: har xml-filene som viser topp menu-baren i applikasjonen vår slik at det gjor lettere for brukeren å navigere de store og fleste hoved skjermer i applikasjonen.
- values: har forskjellige filer som kan gjenbrukes f.eks. *strings*.xml: brukes for å gi et variabel for hver tekst blir brukt i applikasjonen vår slik at de kan gjenbrukes i flere plasser og i tillegg kan de brukes til å oversette appen til flere språk. En annen file er

styles og *colors* som har variabler til styling av forskjellige android komponenter brukes i applikasjonen vår

3.1 Biblioteker

3.1.1 Server

Maven: er et verktøy som vi har brukt til å bygge og administrere prosjektet vårt i server-delen. Ved hjelp av maven kunne vi lett organisere hvordan prosjektet skal bygges opp, hvilke avhengigheter prosjektet har. Kjernekonsept for maven er *pom-filen* (Project Object Model) som inneholder alle informasjon, konfigurasjoner og avhengigheter relatert prosjektet samt ved hvilken versjon vi ønsket å bruke. I tillegg til *pom-filen* har også maven Avhengigheter (dependencies) som sier om alle eksterne java-biblioteker som kreves for å bygge prosjektet. Maven har også *Bygg plugins* som brukes til å utføre et spesifikt mål og all plugins blir lagt til i *pom-filen*. Maven har noen standard plugins som man kan bruker. Plugins kan også implementeres og legges til som vi har gjort i prosjektet vårt.

Quarkus: er et full-stack native java-rammeverk laget for Java Virtual Machines og innfødt kompilering. Den optimaliserer java spesielt for containere og lar det bli effektivt alternativ for serverløs. Som nevnt tidligere brukte vi maven til å bygge prosjektet, og dette bruker vi sammen med quarkus for å kjøre og å kompilere serveren vår

Agroal: er en database connector som tillater å endre på konfigurasjon ved kjøretid, og har førsteklasses integrasjon med de andre komponentene i quarkus, for eksempel sikkerhet, transaksjons styringskomponenter

H2: er en innebygd database og kan kjøre som en server, basert på en fil eller lagre hele data i midlertidig minnet. H2 database har vi brukt i server-testene der vi tester all endepunktene våre. H2 tillater oss å kjøre en sql-script fil før hver test med testdata slik at ingen av originale data blir påvirket eller ødelagt av testene

Junit5: er et Java enhetstesting rammeverk. Testene til til alle endepunktene våre i serveren er satt og blir kjørt via junit som ga oss muligheten til å kjøre repeterbare og automatiske tester.

3.1.2 Klient

Mapbox: en leverandør av online kart. I prosjektet bruker vi Mapbox sitt bibliotek Maps SDK for Android. Dette biblioteket bruker vi til alle kartkomponenter i applikasjonen, og det lar oss blant annet plassere markører og tegne ruter mellom punkter på kartene. Mapbox sine kart kan også skreddersys etter utviklernes behov, slik at vi for eksempel kan bruke sjøkart i applikasjonen.

JUnit4: et rammeverk for enhetstester i Java. På klientsiden av systemet vårt brukte vi JUnit4 sammen med Espresso som rammeverk. JUnit4 muliggjorde for oss å fortelle koden hvilken rekkefølge testene kjøres og hva som skal skje rett før eller etter hver test eller alle testene. Dette blir implementert ved hjelp av JUnit4 med testmetoder som kan markeres med “BeforeEach”, “@Ruler” “@FixMethodOrder()”.

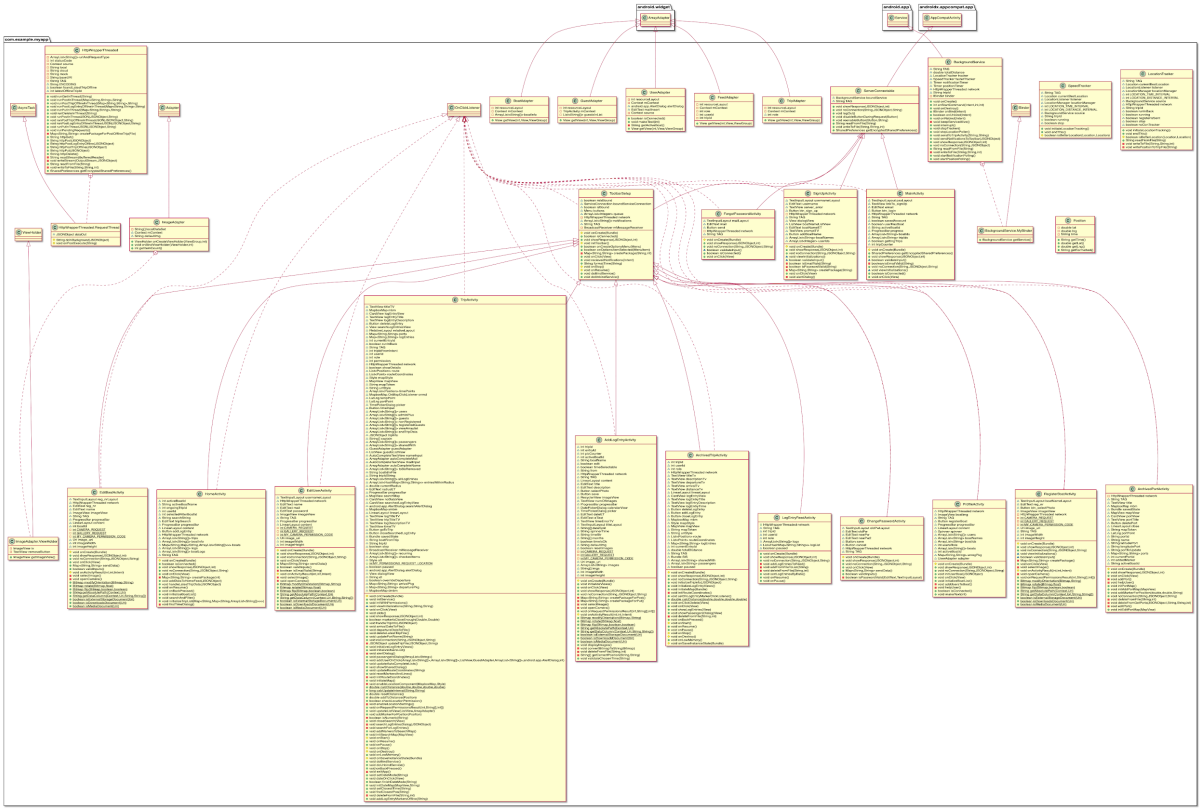
Espresso: et API som brukes til testing av brukergrensesnittet i Android-applikasjoner. Ved hjelp av Espresso kunne vi simulere interaksjoner brukeren har med forskjellige komponenter i applikasjonen. I tillegg kunne vi bruke async-metoder i testene, som vil si å vente på at en komponent har blitt initialisert før vi sjekker verdiene som ligger inne eller under.

CardView og RecyclerView: to API-er som brukes til styling av komponenter flere steder i prosjekter. CardView brukes blant annet til visning av logginnlegg, mens RecyclerView brukes til visning av bilder.

Security: et API som brukes til kryptering av mail og passord i SharedPreferences (mer om det i kap. 7).

4. Klassediagram

4.1 Klassediagram over klient



Figur 5.1 klassediagram over klient

Detaljene er litt vanskelige å se så diagrammet ligger som et vedlegg i innleveringen. Inndelingen av klassene kan sees på som tre deler:

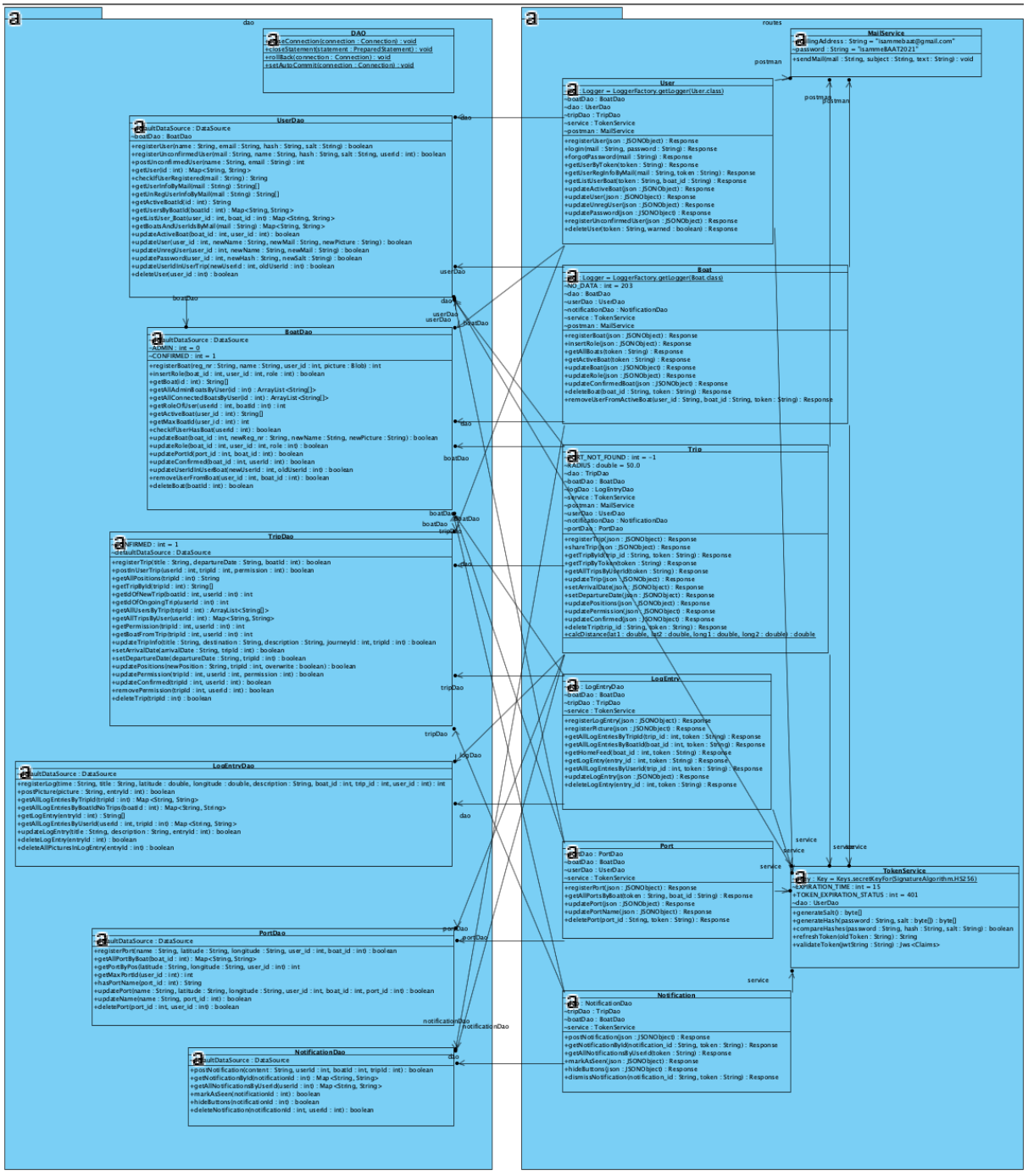
1. Aktiviteter som kjører før brukeren logger inn
2. Aktiviteter som kjører etter brukeren har logget inn
3. Hjelpeklasser

HTTPWrapper er en hjelpeklasse som tar seg koblingen mot server i egne tråder. For at den skal kunne sende responsen til riktig aktivitet lagres kilden som kontekst, og responsen blir sendt til en metode som alle aktivitetene arver. Aktivitetene som kjører før brukeren logger inn arver fra klassen ServerConnectable og aktivitetene som kjører etter brukeren logger inn arver fra klassen ToolbarSetup som igjen arver fra ServerConnectable. Slik kan HTTPWrapper bare kalle en metode som hver aktivitet har sin egen implementasjon av.

Diagrammet viser også at de fleste aktivitetene implementerer onClickListener for å kunne føre alle trykk på elementer i den tilhørende xml filen til en metode som kalles onClick(). Diagrammet viser også alle de egendefinerte adapterne våre som arver fra ArrayAdapter.

Noen av hjelpeklassene som er verdt å nevne er Position som brukes for å lagre data om et punkt inneholde både tidspunkt, breddegrad og lengdegrad, SpeedTracker (oppdateres kjapt for å vise fart) og LocationTracker (oppdateres litt senere for å lagre punkter på ruta) er klasser som tar seg av sporingen ved hjelp av GPS. BackgroundService er en service klasse som holder styr på alt som skal kjøres i bakgrunnen. Her opprettes objekter av de to Trackerklassene, og opptil to andre tråder kjøres for å hente varsler fra databasen og posisjoner om klienten ikke er den som startet turen. Da må klienten nemlig hente punktene fra serveren når de blir lagret.

4.2 Klassediagram over serveren



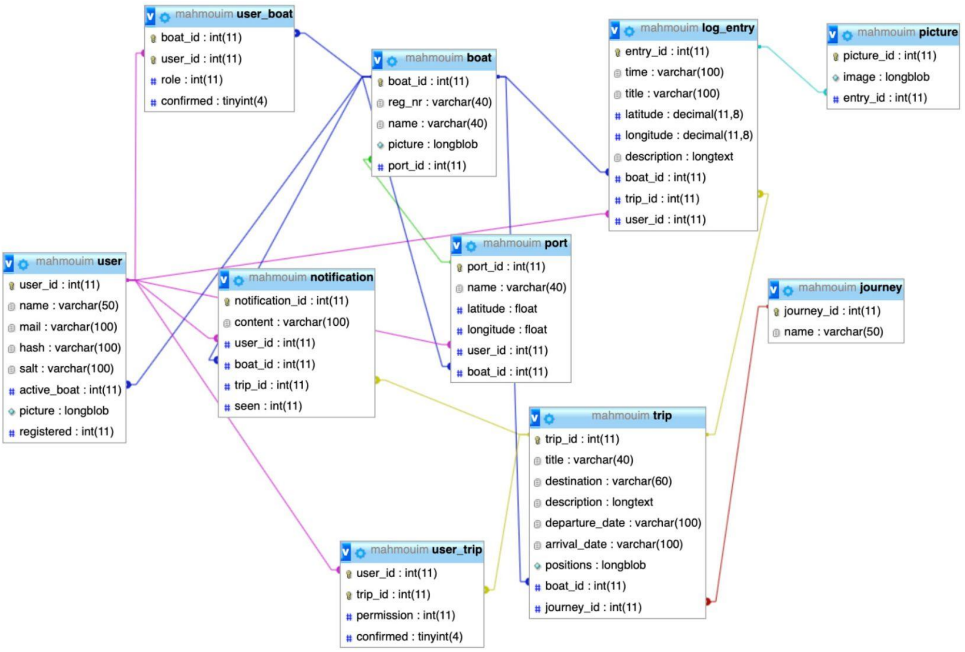
Figur 5.2 klassediagram over server

Det er litt vanskelig å se detaljene på diagrammet her så det er også lagt ved som vedlegg i innleveringen. I diagrammet ser vi koblingene mellom klassene på serveren. Forespørslene fra klient ankommer først den passende ruten i routes mappen. Noen av rutene injecter hjelpeklassene MailService og TokenService. MailService sørger for å sende mail til

brukerne mens TokenService inneholder mange metoder for å validere, oppdatere og opprette token.

Rutene oppretter objekter av dao variantene de trenger og ber dem utføre spørringene til databasen. Alle rutene oppretter et objekt av sin tilhørende dao-klasse, men noen trenger også å kjøre metoder som ligger i andre dao-klasser og må derfor også opprette varianter av den typen. Dette skjer for eksempel i Boat-ruten hvor vi oppretter et dao-objekt av typen BoatDao, men trenger også å hente informasjon om brukeren og oppretter derfor et dao-objekt av typen UserDao også. Alle dao-klassene arver fra en hovedklasse ved navn DAO som inneholder funksjoner alle underklassene må ha for å commite, utføre rollback og lukke forbindelser.

5. Databasemodell



Figur 5.1 databasemodell

Databasen i relasjonsmodell form er som følger:

- user (user_id, name, mail, hash, salt, picture, registered, active_boat*)
- boat (boat_id, reg_nr, name, picture, port_id*)
- user_boat ((user_id*, boat_id*), role, confirmed)
- trip (trip_id, title, destination, description, departure_date, arrival_date, positions, boat_id*, journey_id*)
- user_trip ((trip_id*, user_id*), permission, confirmed)
- log_entry (entry_id, time, title, latitude, longitude, description, boat_id*, trip_id*, user_id*)
- picture (picture_id, image, entry_id*)
- port (port_id, name, type, latitude, longitude, user_id*, boat_id*)
- notification (notification_id, content, seen, user_id*, boat_id*, trip_id*)
- journey (journey_id, name)

I database ER-diagram og modelleringen viser vi hvilke koblinger som gjøres mellom de ulike entitetene i databasen vår. Tar man utgangspunkt i de første tre entitetene ser man at vi har to relasjoner mellom *user-entiteten* og *boat-entiteten*. Den første er mange-til-mange som vil si at en bruker kan eie, være mannskap eller gjest i flere båter, og samtidig kan båten bli eid av flere brukere, ha flere mannskap eller gjester. Derfor har vi hjelpe tabellen *user_boat-entiteten*. Den andre relasjonen er en-til-en som sier om hvilken båt som er i bruk på et gitt tidspunkt, og som kan være null hvis brukeren ikke har noe aktiv båt.

I tillegg til *boat-entiteten* ser vi at *user-entiteten* knyttet til *trip-entiteten gjennom hjelpetabellen user_trip-entiteten*. Hjelpetabellen har permission som kan ha tre forskjellige verdier. 0 som forteller oss om brukeren som har startet turen, 1 personer som har mulighet til å lese og å skrive logginlegg og den siste er 2 som kan kun se. Vi ser også at vi har en confirmed kolonne i både *user_boat-entiteten og user_trp-entiteten*. Dette brukes til sjekke om brukeren har akseptert invitasjonen eller ikke.

Videre ser vi på *log_entry-entiteten* har relasjoner til tre forskjellige entiteter. Den første er mange-til-en med *user-entiteten* som sier at en bruker kan skrive flere logg, mens et logginlegg kan bli skrevet av kun en bruker. Den andre er en-til-en med både *boat-entiteten* og *trip-entiteten* som forteller om hvilken trip og båt tilhører et logginlegg til. trip_id kan i dette tilfelle være null, som betyr at brukeren har skrevet et logginlegg uten trip. Grunnen til at vi har bestemt oss å ha både trip_id og boat_id i *log_entry-entiteten* er at et logginlegg ikke kan eksistere uten båt og trip eller bare båt. Vi ser også at *log_entry-entiteten* har en-til-mange relasjon med picture som betyr at et logginnlegg kan ha null eller flere bilder.

Vi har også to relasjoner mellom *port-entiteten* og *boat-entiteten.* Den første er mange til mange som betyr at båten kan ha flere havner og motsatt. Den andre er en-til-en som forteller oss om i hvilken havn båten ligger på et tidspunkt. Den siste kan være null dersom brukeren velger å ikke angi noe informasjon om dette. *port-entiteten* har også en-til-en relasjon med *user-entiteten* som forteller om hvilke bruker som eier havna. Dette er implementert i tillegg til relasjonen med *port-entiteten* på grunn av det som sagt i starten at en båt kan blir eid av flere. Derfor har vi med user_id i *port-entiteten* for å skille mellom hvilken bruker som eier havna.

Til slutt ser vi på *notification-entiteten* som er en hjelpetabell mellom de store tre entitetene vi har *user-entiteten, boat-entiteten og trip-entiteten,* som brukes til kommunikasjon delen mellom brukere og varsle-systemet i appen. Vi har også en siste en-til-mange relasjon mellom *journey-entiteten* og *trip-entiteten. Den siste relasjonen betyr at flere turer kan bli samlet inn i en stor tur som f.eks sommertur. I applikasjonen har vi ikke brukt den relasjonen fordi den var lavt prioritert etter ønske fra Produkt-eieren, men vi tenkte å ta med for videreutvikling og vise hvordan vi har planlagt hele systemet.*

1 boat

Creation: Apr 30, 2021 at 03:11 PM
Last update: Apr 30, 2021 at 06:09 PM

Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
boat_id	int(11)		No		auto_increment			
reg_nr	varchar(40)		Yes	NULL				
name	varchar(40)		No					
picture	longblob		Yes	NULL				
port_id	int(11)		Yes	NULL		-> port.port_id ON UPDATE CASCADE ON DELETE SET_NULL		

2 journey

Creation: Apr 30, 2021 at 03:11 PM
Last update: Apr 30, 2021 at 03:11 PM

Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
journey_id	int(11)		No		auto_increment			
name	varchar(50)		No					

3 log_entry

Creation: Apr 30, 2021 at 03:11 PM
Last update: Apr 30, 2021 at 05:53 PM

Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
entry_id	int(11)		No		auto_increment			
time	varchar(100)		No					
title	varchar(100)		No					
latitude	decimal(11, 8)		Yes	NULL				
longitude	decimal(11, 8)		Yes	NULL				
description	longtext		Yes	NULL				
boat_id	int(11)		No			-> boat.boat_id ON UPDATE CASCADE ON DELETE CASCADE		
trip_id	int(11)		Yes	NULL		-> trip.trip_id ON UPDATE CASCADE ON DELETE CASCADE		
user_id	int(11)		Yes	NULL		-> user.user_id ON UPDATE CASCADE ON DELETE SET_NULL		

4 notification

Creation: Apr 30, 2021 at 03:11 PM
Last update: Apr 30, 2021 at 06:09 PM

Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
notification_id	int(11)		No		auto_increment			
content	varchar(100)		No					
user_id	int(11)		No			-> user.user_id ON UPDATE CASCADE ON DELETE CASCADE		
boat_id	int(11)		Yes	NULL		-> boat.boat_id ON UPDATE CASCADE ON DELETE CASCADE		
trip_id	int(11)		Yes	NULL		-> trip.trip_id ON UPDATE CASCADE ON DELETE CASCADE		
seen	int(11)		No					

5 picture

Creation: Apr 30, 2021 at 03:11 PM
Last update: Apr 30, 2021 at 05:53 PM

Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
picture_id	int(11)		No		auto_increment			
image	longblob		No					
entry_id	int(11)		No			-> log_entry.entry_id ON UPDATE CASCADE ON DELETE CASCADE		

6 port

Creation: Apr 30, 2021 at 06:23 PM
Last update: Apr 30, 2021 at 06:27 PM

Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
port_id	int(11)		No		auto_increment			
name	varchar(40)		No					
latitude	float		No					
longitude	float		No					
user_id	int(11)		No			-> user.user_id ON UPDATE CASCADE ON DELETE CASCADE		
boat_id	int(11)		Yes	NULL		-> boat.boat_id ON UPDATE RESTRICT ON DELETE RESTRICT		

7 trip

Creation: Apr 30, 2021 at 03:11 PM
Last update: Apr 30, 2021 at 03:11 PM

Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
trip_id	int(11)		No		auto_increment			
title	varchar(40)		No					
destination	varchar(60)		Yes	NULL				
description	longtext		Yes	NULL				
departure_date	varchar(100)		No					
arrival_date	varchar(100)		Yes	NULL				
positions	longblob		No					
boat_id	int(11)		No			-> boat.boat_id ON UPDATE CASCADE ON DELETE CASCADE		
journey_id	int(11)		Yes	NULL		-> journey.journey_id ON UPDATE CASCADE ON DELETE CASCADE		

8 user

Creation: Apr 30, 2021 at 03:11 PM
Last update: Apr 30, 2021 at 06:37 PM

Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
user_id	int(11)		No		auto_increment			
name	varchar(50)		No					
mail	varchar(100)		Yes	NULL				
hash	varchar(100)		No					
salt	varchar(100)		No					
active_boat	int(11)		Yes	NULL		-> boat.boat_id ON UPDATE CASCADE ON DELETE SET_NULL		
picture	longblob		Yes	NULL				
registered	int(11)		No					

10 user_trip

Creation: Apr 30, 2021 at 03:11 PM
Last update: Apr 30, 2021 at 03:11 PM

Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
user_id	int(11)		No			-> user.user_id ON UPDATE CASCADE ON DELETE CASCADE		
trip_id	int(11)		No			-> trip.trip_id ON UPDATE CASCADE ON DELETE CASCADE		
permission	int(11)		Yes	NULL				
confirmed	tinyint(4)		Yes	0				

6. Server-tjenester

Til alle endepunktene vi har i systemet vårt sender vi to forskjellige URI-er avhengig av hvilken fase vi er i. Den første er lokal og brukes i utviklingsfase, og den andre er til google cloud serveren og brukes til testing og når applikasjonen skal brukes.
Nettadressene som serveren får fra klienten er som følgende:

1. Lokalt : http://10.0.2.2:8080/
2. Cloud: https://strange-mind-305617.ew.r.appspot.com/

I alle tabellene under 6. avsnittet skal vi bruke baseURI som referer til de to nettadressene skrevet over. I tillegg starter alle nettadresser med https selv om at de ikke gjør det i utviklingsfasen, men vi skriver det for å vise hvordan endepunktene ser ut når appen faktisk brukes.

6.1 User.java

Metodenavn	**POST registerUser()**
url	https://strange-mind-305617.ew.r.appspot.com/users
Bruk	// Registrere en bruker

body	{
	"navn": "user",
	"mail": "user@user.no",
	"password": "Test12",
	}

Vellykket **Respons**	Status:201	innhold: {"token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "tag": "registerUser"}

	Status: 300	innhold: {
		
		"liste over båter som brukeren er tilknyttet til som gjest
		
		}

Error Response	Status:403	innhold: error: "Brukeren finnes fra før"
	Status:403	innhold: error: "Forbudt"
	code: 405	innhold: error: "server error"

Metodenavn	**GET login()**
url	https://strange-mind-305617.ew.r.appspot.com/users/login/{mail}/{password}

Bruk	// Logg inn	
body	Tom, data blir sendt som parametere i uri	
Vellykket **Respons**	Status:200	innhold: {"token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "user_id":"1", "name":"user", "tag":"login" }
Error respons	status:400	innhold: error : “Feil ved anmodning”
	status:405	innhold:error: “ikke registrert”

Vi hadde sendt flere tilbakemelding med forskjellige statuskode og behandlet dem i klientside, men produkteieren mente at vi ikke skal gi mye informasjon på grunn av sikkerhet. Derfor har vi valgt å sende bar statuskode på det som er nødvendig.

Metodenavn	**GET forgotPassword()**
url	https://strange-mind-305617.ew.r.appspot.com/users/forgot_password/{mail}

Bruk	// Tilbakestille passord og sende et nytt passord til brukeren	
body	Tom, data blir sendt som parameter i uri	
Vellykket **Respons**	Status:200	innhold: (Vi sender ingen melding, men vi sender en mail til brukeren med informasjon som trenges
Error respons	status:404	innhold: error : “Ikke funnet”

Metodenavn	**GET getUserByToken()**	
url	https://strange-mind-305617.ew.r.appspot.com/users/{token}	
Bruk	// Hente info om en bruker til å vise dem i profil og i endre brukerinfo	
body	Tom, data blir sendt som parameter i uri	
Vellykket **Respons**	Status:200	innhold: { "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM

		"name": "user", "mail": "user@user.no", "token": "newToken" "picture": "Blob",}
Error respons	status:404	innhold: error : “Uaturoisert”

Metodenavn	**GET getUserRegInfoByMail()**	
url	https://strange-mind-305617.ew.r.appspot.com/users/reg_info/{mail}/{token}	
Bruk	// Hente info om en bruker som er tilknyttet til en eller flere båter i systemet.	
body	Tom, data blir sendt som parameter i uri	
Vellykket **Respons**	Status:200	innhold: { "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM, "regInfo": "registered | not registered | not confirmed",

		"mail": "user@user.no", "tag": "getUserRegInfoByMail",}
Error respons	status:401	innhold: error : “Uautorisert”

Metodenavn	**GET getListUserBoat()**	
url	https://strange-mind-305617.ew.r.appspot.com/users/user_boat/{boat_id}/{token}	
Bruk	// Henter alle brukere som er tilknyttet en båt med hvilken relasjon de har for å vise dem i min båt side i applikasjonen Rolle 0 betyr admin, 1 betyr fast tilknyttet til båt f.eks familie og 2 betyr gjest på båten	
body	Tom, data blir sendt som parameter i uri	
Vellykket **Respons**	Status:200	innhold: { "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "user #0": "1¤user¤user@user.no¤0", "boat_id": "1",

		"tag": "getListuserBoat",}
Error respons	status:401	innhold: error : “Uautorisert”
	status:404	innhold: error : “Ikke funnnet”

Metodenavn	**GET updateActiveBoat()**	
url	https://strange-mind-305617.ew.r.appspot.com/users/active_boat/{token}	
Bruk	// Oppdaterer hvilken båt som er i bruk nå ut i fra valget fra brukeren, automatisk hvis båten var den første for brukeren eller sette den siste registrerte båten som active båt hvis brukeren har slettet en båt	
body	{ "boat_id": "1", }	
Vellykket **Respons**	Status:200	innhold: { "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "user #0": "1¤user¤user@user.no¤0", "boat_id": "1",

		"tag": "getListuserBoat",}
Error respons	status:401	innhold: error : “Uautorisert”
	status:404	innhold: error : “Ikke funnnet”

Metodenavn	**PUT getListUserBoat()**	
url	https://strange-mind-305617.ew.r.appspot.com/users/user_boat/{boat_id}	
Bruk	// Henter alle brukere som er tilknyttet en båt med hvilken relasjon de har for å vise dem i min båt side i applikasjonen Rolle 0 betyr admin, 1 betyr fast tilknyttet til båt f.eks familie og 2 betyr gjest på båten	
body	"token":"string som inneholder tokenet”,	
Vellykket **Respons**	Status:200	innhold: {"token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM

		"updated": "boolean type", "tag": "updateActiveBoat",}
Error respons	status:401	innhold: error : “Uautorisert”
	status:404	innhold: error : “Ikke funnnet”
	status:405	innhold: error : “Intern tjenerfeil”

Metodenavn	**PUT updateuser()**
url	https://strange-mind-305617.ew.r.appspot.com/users/users
Bruk	// Oppdaterer brukerinfo til en gitt bruker Rolle 0 betyr admin, 1 betyr fast tilknyttet til båt f.eks familie og 2 betyr gjest på båten
body	{ "token": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "name": "user", "mail": "user@user.no",

	"picture": "Blob", "password": "Test12",}	
Vellykket Respons	Status:200	innhold: { "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "updated": "boolean type", "tag": "updatedUser",}
Error respons	status:400	innhold: error : "Feil ved anmodning"
	status:401	innhold: error : "Uautorisert"
	status:403	innhold: error : "Forbudt"
	status:405	innhold: error : "Intern tjenerfeil"

Metodenavn	**PUT updateUnregUser()**
url	https://strange-mind-305617.ew.r.appspot.com/users/unreg
Bruk	// Admin endrer mail eller navn til en bruker tilknyttet til båten som ikke har konto

body	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "name": "gjest", "mail": "gjest@gjest.no", "user_id": "1", "password": "Test12",}	
Vellykket **Respons**	Status:200	innhold: { "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "updated": "boolean type", "tag": "updateUnreguser",}
Error respons	status:400	innhold: error : “Feil ved anmodning”
	status:401	innhold: error : “Uautorisert”
	status:405	innhold: error : “Intern tjenerfeil”

Metodenavn	**PUT updatePassword()**

url	https://strange-mind-305617.ew.r.appspot.com/users/password	
Bruk	// Bytter passord til en gitt bruker	
body	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "password": "Test12" "newPassword": "Newpassword12",}	
Vellykket **Respons**	Status:200	innhold: { "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey JlbWFpbCI6ImVtYWlsQGVtYWlsLmNvbSIsImlhdC I6MTU3OTc4MTM0NywiZXhwIjoxNTc5NzgzMTQ3fQ .MyEWVMfkrkwTImbkYUd6BwRL4UoBxDQM7gUL9kf z43Q", "updated": "boolean type" "tag": "updatePassword",}
Error respons	status:400	innhold: error : “Feil ved anmodning”

	status:401	innhold: error : “Uautorisert”
	status:403	innhold: error : “Forbudt”
	status:405	innhold: error : “Intern tjenerfeil”

Metodenavn	**PUT registerUnconfirmedUser()**	
url	https://strange-mind-305617.ew.r.appspot.com/users/unconfirmed	
Bruk	// Som nevnt tidligere, tillater vi i systemet brukere som ikke har en konto. Et eksempel på det er barna til båteieren som har relasjon til båt og ønsker å lage data slik at de kan tilknytte det mot den nye kontoen de lagrer i framtida. Denne tilknytningen skjer via dette endepunktet	
body	{ "name": "user2", "mail": "use2r@user.user" "password": "Test12", "userIds": "{1,2,3…. liste over alle brukere som ikke har konto }",}	
Vellykket Respons	Status:200	innhold: { }

Error respons	status:400	innhold: error : “Feil ved anmodning”
Error respons	status:500	innhold: error : “Intern tjenerfeil”

Metodenavn	**DELETE deleteUser()**	
url	https://strange-mind-305617.ew.r.appspot.com/users/{warned}/(token}	
Bruk	// Som nevnt tidligere, tillater vi i systemet brukere som ikke har en konto. Et eksempel på det er barna til båteieren som har relasjon til båt og ønsker å lage data slik at de kan tilknytte det mot den nye kontoen de lagrer i framtida. Dette tilknytning skjer via dette endepunktet	
body	Tom, data blir sendt som parameter i uri	
Vellykket Respons	Status:200	innhold: { }
Error respons	status:500	innhold: error : “Intern tjenerfeil”

6.2 Boat.java

Metodenavn	**POST registerBoat()**	
url	https://strange-mind-305617.ew.r.appspot.com/boats	
Bruk	// Registrer en ny båt	
body	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "name": "saarlandet", "reg_nr": "TR0123", "picture": "BLOB", "password": "Test12",}	
Vellykket **Respons**	Status:201	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "activeBoatSet": "boolean" // sendes bare når det er den første båten til brukeren, dvs at

		båten blir satt som aktiv båt i user entity i mysql // "tag": "registerBoat", "boat_id": "2",}
Error respons	status:400	innhold: error : “Feil ved anmodning”
	status:401	innhold: error : “Uautorisert”
	status:500	innhold: error : “Intern tjenerfeil”

Metodenavn	**POST insertRole()**
url	https://strange-mind-305617.ew.r.appspot.com/boats /role
Bruk	// Legger til brukere i en gitt båt, vi har 3 forskjellige role som er nevnt over i getLisUserBoat metode under 6.1
body	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "role": "0,1 eller 2",

	"mail": "admin@admin.no", "boat_id": "1",}	
Vellykket **Respons**	Status:200	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "created": "boolean" "tag": "insertRole",}
Error respons	status:401	innhold: error : “Uaturisert”
	status:500	innhold: error : “Intern tjenerfeil”

Metodenavn	**GET getAllBoats()**
url	https://strange-mind-305617.ew.r.appspot.com/boats /{token}
Bruk	// Henter alle båtene til en bruker
body	Tom, data som trenges er bruker id og den blir sendt med inne tokentet

Vellykket **Respons**	Status:200	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "created": "boolean" "tag": "getAllBoats", "boat #0": "boat id+ name+ reg_nr...", "boat #1": "boat id+ name+ reg_nr...",}
Error respons	status:401	innhold: error : “Uautorisert”

Metodenavn	**GET getActiveBoat()**
url	https://strange-mind-305617.ew.r.appspot.com/boats /active_boat/{token}
Bruk	// Henter den aktiv båten til en gitt bruker og viser for å vise info om den båten i mine båter-side
body	Tom, data som trenges er bruker id og den blir sendt med inne tokentet

Vellykket **Respons**	Status:200	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "boat_id": "1" "name": "Saarlandet" "picture": "BLOB" "reg_nr": "TR012" "role": "0" "tag": "getActiveBoat",}
Error respons	status:401	innhold: error : “Uautorisert”

Metodenavn	**PUT updateBoat()**
url	https://strange-mind-305617.ew.r.appspot.com/boats
Bruk	// Redigerer den aktiv båten til en gitt bruker
body	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ

	4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "boat_id": "1" "name": "Saarlandet" "picture": "BLOB" "reg_nr": "TR012" }	
Vellykket **Respons**	Status:200	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "updated": "boolean" "tag": "updateBoat",}
Error respons	status:401	innhold: error : “Uautorisert”
	status:500	innhold: error : “Intern tjenerfeil”

Metodenavn	**PUT updateRole()**
url	https://strange-mind-305617.ew.r.appspot.com/boats/role
Bruk	// Redigerer på relasjon mellom båten og brukeren

body	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "boat_id": "1" "user_id": "1" "role": "1" }	
Vellykket Respons	Status:200	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "updated": "boolean" "tag": "updateRole",}
Error respons	status:401	innhold: error : “Uautorisert”
	status:500	innhold: error : “Intern tjenerfeil”

Metodenavn	**PUT updateConfirmedBoat()**
url	https://strange-mind-305617.ew.r.appspot.com/boats/user_confirmed

Bruk	// Setter inne user_boat entitet at brukeren har akseptert å bli lagt til i en gitt båt	
body	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "boat_id": "1" }	
Vellykket **Respons**	Status:200	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "updated": "boolean" "boat_id": "1" "tag": "updateConfirmedBoat",}
Error respons	status:401	innhold: error : "Uautorisert"
	status:500	innhold: error : "Intern tjenerfeil"

Metodenavn	**DELETE deleteBoat()**
url	https://strange-mind-305617.ew.r.appspot.com/boats/{boat_id}/{token}

Bruk	// Sletter en båt av systemet	
body	Tom, all data som trenges blir sendt med som parametere i uri	
Vellykket **Respons**	Status:200	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "activeBoatAfterDelete": "boolean" "tag": "deleteBoat",}
Error respons	status:401	innhold: error : “Uautorisert”
	status:500	innhold: error : “Intern tjenerfeil”

Metodenavn	**DELETE removeUserFromActiveBoat()**
url	https://strange-mind-305617.ew.r.appspot.com/boats/user_boat/{user_id }/{boat_id}/{token}
Bruk	// Sletter en tilknyttede bruker til en båt
body	Tom, all data som trenges blir sendt med som parametere i uri

Vellykket **Respons**	Status:200	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "removed": "boolean" "tag": "removeUserFromActiveBoat",}
Error respons	status:401	innhold: error : “Uautorisert”
	status:500	innhold: error : “Intern tjenerfeil”

6.3 Port.java

Metodenavn	**POST registerPort()**
url	https://strange-mind-305617.ew.r.appspot.com/ports
Bruk	// Registrer en ny havn, enten manuelt eller automatisk basert på avgangs og ankomst posisjon til en tur
body	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM

	"name": "Trondheim kai", "latitude": "44.45", "longitude": "10.23",}	
Vellykket **Respons**	Status:201	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "registered": "boolean true" "tag": "registerPort", "boat_id": "2",}
Error respons	status:400	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "registered": "boolean false" "tag": "registerPort", "boat_id": "2",}
	status:401	innhold: error : “Uautorisert”
	status:500	innhold: error : “Intern tjenerfeil”

Metodenavn	**GET getAllPortByBoat()**	
url	https://strange-mind-305617.ew.r.appspot.com/ports/{boat_id)/{token}	
Bruk	// Henter alle havner tilknyttet en båt for å vise dem på et kart i applikasjonen	
body	Tom, data som trenges blir sendt som parameter i uri	
Vellykket **Respons**	Status:200	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "tag": "getAllPortsByBoat",}
Error respons	status:401	innhold: error : “Uautorisert”

Metodenavn	**PUT updatePort()**
url	https://strange-mind-305617.ew.r.appspot.com/ports
Bruk	// Redigerer en havn

body	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "name": "Trondheim kai", "port_id": "1", "latitude": "44.45", "longitude": "10.23",}	
Vellykket **Respons**	Status:200	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "updated": "boolean: true" "tag": "updatePort",}
Error response	status: 400	innhold: error : “Feil ved anmodning”
	status:401	innhold: error : “Uautorisert”

Metodenavn	**PUT updatePort()**

url	https://strange-mind-305617.ew.r.appspot.com/ports/name	
Bruk	// Redigerer et navn til en havn, og vi bruker ikke den forrige endepunktet er at vi ikke har tilgang til posisjonen der vi bruker den	
body	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "name": "Trondheim kai", "port_id": "1",}	
Vellykket **Respons**	Status:200	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "updated": "boolean: true" "tag": "updatePort",}
Error response	status: 400	innhold: error : “Feil ved anmodning”
	status:401	innhold: error : “Uautorisert”

Metodenavn	**DELTE deletePort()**	
url	https://strange-mind-305617.ew.r.appspot.com/ports/{port_id}/{token}	
Bruk	// Redigerer en havn	
body	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "name": "Trondheim kai", "port_id": "1", "latitude": "44.45", "longitude": "10.23",}	
Vellykket **Respons**	Status:200	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "updated": "boolean: true" "tag": "updatePort",}
Error response	status: 400	innhold: error : “Feil ved anmodning”
	status:401	innhold: error : “Uautorisert”

6.4 LogEntry.java

Metodenavn	**POST registerLogEntry()**	
url	https://strange-mind-305617.ew.r.appspot.com/log_entries	
Bruk	// Registrer et nytt logginnlegg	
body	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "trip_id": "1", "boat_id": "1", "tilte": "seilene er satt", "description": "Vi satt upper seil og slår av motoren", "latitude": "44.45", "longitude": "10.23",}	
Vellykket **Respons**	Status:201	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "entry_id": "1"

		"tag": "registerLogEntryt",}
Error respons	status:401	innhold: error : “Uautorisert”
	status:500	innhold: error : “Intern tjenerfeil”

Metodenavn	**POST regiserPicture()**	
url	https://strange-mind-305617.ew.r.appspot.com/log_entries/picture	
Bruk	// Registrer et eller flere bilder til et logginnlegg, og grunnen til at vi ikke har både bilder og logginnlegget i same endepunktet er at vi tillater flere bilder og har en egen entity for bildene i database som har et logginnlegg-id som refererer til logginnlegget bildene tilhører til. Den fremmednøkkelen kan ikke være null	
body	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "image": "1", "entry_id": "1",}	
Vellykket Respons	Status:201	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOj

		E2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "registered": "boolean" "tag": "registerPicture",}
Error respons	status:401	innhold: error : “Uautorisert”
	status:500	innhold: error : “Intern tjenerfeil”

Metodenavn	**GET getAllLogEntriesByTripId()**	
url	https://strange-mind-305617.ew.r.appspot.com/log_entries/trip_id/{trip_id}/{token}	
Bruk	// Henter alle logginnleggene som blir lagt i en spesifikk tur	
body	Tom, all data som trenges blir sendt med uri som parameter	
Vellykket **Respons**	Status:201	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "user_id": "1"

		"tag": "getAllLogEntriesByTripId",}
Error respons	status:401	innhold: error : “Uautorisert”
	status:500	innhold: error : “Intern tjenerfeil”

Metodenavn	**GET getAllLogEntriesByTripId()**	
url	https://strange-mind-305617.ew.r.appspot.com/log_entries/boat_id/{boat_id}/{token}	
Bruk	// // Henter alle logginnleggene som blir lagt i en spesifikk båt uten å ha startet en tur	
body	Tom, all data som trenges blir sendt med uri som parameter	
Vellykket **Respons**	Status:201	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "user_id": "1" "tag": "getAllLogEntriesByTripId",}
Error respons	status:401	innhold: error : “Uautorisert”

	status:500	innhold: error : “Intern tjenerfeil”

Metodenavn	**GET getHomeFeed()**	
url	https://strange-mind-305617.ew.r.appspot.com/log_entries/{boat_id}/{token}	
Bruk	// // Henter alle data som er tilknyttet en båt f.eks alle turer, logginnleggene til turer eller båten, alle havner og brukere	
body	Tom, all data som trenges blir sendt med uri som parameter	
Vellykket **Respons**	Status:201	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "user_id": "1" "tag": "getHomeFeed",}
Error respons	status:401	innhold: error : “Uautorisert”
	status:500	innhold: error : “Intern tjenerfeil”

Metodenavn	**GET getLogEntry()**	
url	https://strange-mind-305617.ew.r.appspot.com/entry_id/{entry_id}/{token}	
Bruk	// Henter info om et spesifikt logginnlegg	
body	Tom, all data som trenges blir sendt med uri som parameter	
Vellykket **Respons**	Status:200	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "trip_id": "1", "resetImages": "boolean", "tilte": "motoren kjører igjen", "description": "Vi satt upper seil og slår av motoren", "latitude": "44.45", "longitude": "10.23", "image #0": "Base64 string", "tag": "getLogEntry",}
Error respons	status:401	innhold: error : “Uautorisert”

	status:404	innhold: error : “Ikke funnet”

Metodenavn	**GET getAllLogEntriesByUserId()**	
url	https://strange-mind-305617.ew.r.appspot.com/search/{trip_id}/{token}	
Bruk	// Henter all logginnleggne brukeren har for å vise det som er i nærheten av brukeren på kartet innenfor et gitt radius som brukeren bestemmer på klientside	
body	Tom, all data som trenges blir sendt med uri som parameter	
Vellykket **Respons**	Status:200	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "entry #0, #1...": {liste over alle logginnlegg} "tag": "getAllLogEntriesByUserId",}
Error respons	status:401	innhold: error : “Uautorisert”

Metodenavn	**PUT updateLogEntry()**	
url	https://strange-mind-305617.ew.r.appspot.com/log_entries	
Bruk	// Redigerer et logginlegg brukeren har gitt id til logginlegget	
body	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "entry_id": "1", "tilte": "motoren har stoppet", "resetImages": "boolean", "description": "Vi satt upper seil igjen siden motoren ikke fungerer", "latitude": "44.45", "longitude": "10.23",}	
Vellykket **Respons**	Status:201	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "entry_id": "1", "updated": "boolean",

		"tag": "updateLogEntry",}
Error respons	status:401	innhold: error : “Uautorisert”
	status:500	innhold: error : “Intern tjenerfeil”

Metodenavn	**DELETE deleteLogEntry()**	
url	https://strange-mind-305617.ew.r.appspot.com/log_entries/{entry_id}/{token}	
Bruk	// Sletter et logginnlegg gitt id	
body	Tom, data som trenges sendes med i uri som parameter	
Vellykket Respons	Status:201	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "entry_id": "1" "tag": "deleteLogEntry",}
Error respons	status:401	innhold: error : “Uautorisert”
	status:500	innhold: error : “Intern tjenerfeil”

6.5 Trip.java

Metodenavn	**POST registerTrip()**	
url	https://strange-mind-305617.ew.r.appspot.com/trips	
Bruk	// Registrer en ny tur	
body	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "boat_id": "1", "tilte": "seilene er satt", "description": "Vi satt upper seil og slår av motoren", "departure_date": "Thu Feb 18 09:42:52 GMT+01:00 2021",}	
Vellykket **Respons**	Status:201	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "trip_id": "1" "tag": "registerTrip",}

Error respons	status:401	innhold: error : “Uautorisert”
	status:500	innhold: error : “Intern tjenerfeil”

Metodenavn	**POST shareTrip()**	
url	https://strange-mind-305617.ew.r.appspot.com/trips/share	
Bruk	// Deler en tur med en annen bruker	
body	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "boat_id": "1", "registered #0": {"user1, user1..."}, "user #0": "user1, user2 ", // brukere som er registrert, men ikke har en konto "departure_date": "Thu Feb 18 09:42:52 GMT+01:00 2021",}	
Vellykket Respons	Status:201	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM

		"tag": "sahreTrip",}
Error respons	status:401	innhold: error : “Uautorisert”
	status:500	innhold: error : “Intern tjenerfeil”

Metodenavn	**GET getTripById()**	
url	https://strange-mind-305617.ew.r.appspot.com/trips/{trip_id}/{token}	
Bruk	// Henter en spesifikk tur gitt id	
body	Tom, data som trenges blir sendt med i uri som parameter	
Vellykket **Respons**	Status:200	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "trip_id": "1", "boat_id": "1", "journey_id": "1", "role": "1",

		"permission": "1", // forteller om brukeren har leserettighet eller bare skriverettighet "tilte": "motoren har stoppet", "destination": "Oslofjord", "description": "Vi planlegger å seile fra trondheim til Oslofjord uten å bruke motor, "departure_date": "Thu Feb 18 09:42:52GMT+01:002021" ”arrival_date": "Thu Feb 18 09:42:52GMT+01:002021" "latitude": "44.45", "longitude": "44.45", "positions": "LongBlob",} "tag": "getTripById",}
Error respons	status:401	innhold: error : “Uautorisert”
	status:500	innhold: error : “Intern tjenerfeil”

Metodenavn	**GET getTripByToken()**

url	https://strange-mind-305617.ew.r.appspot.com/trips/latest/{token}	
Bruk	// Henter den siste pågående turen en bruker har	
body	Tom, data som trenges blir sendt med i uri som parameter	
Vellykket **Respons**	Status:200	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "trip_id": "1", "boat_id": "1", "journey_id": "1", "role": "1", "permission": "1", // forteller om brukeren har leserettighet eller bare skriverettighet "tilte": "motoren har stoppet", "destination": "Oslofjord", "description": "Vi planlegger å seile fra trondheim til Oslofjord uten å bruke motor, "departure_date": "Thu Feb 18 09:42:52GMT+01:002021"

		”arrival_date": "Thu Feb 18 09:42:52GMT+01:002021" "latitude": "44.45", "longitude": "44.45", "positions": "LongBlob",} "tag": "getTripByToken",}
Error respons	status:401	innhold: error : “Uautorisert”
	status:500	innhold: error : “Intern tjenerfeil”

Metodenavn	**GET getAllTripByUserId()**	
url	https://strange-mind-305617.ew.r.appspot.com/trips	
Bruk	// Henter alle turene til en bruker	
body	Tom, data som trenges blir sendt med i uri som parameter	
Vellykket **Respons**	Status:200	Liste over alle turer brukeren har som og en tur ser sånn ut { ”trip #0”: "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI

		0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "trip_id": "1", "boat_id": "1", "journey_id": "1", "role": "1", "permission": "1", // forteller om brukeren har leserettighet eller bare skriverettighet "tilte": "motoren har stoppet", "destination": "Oslofjord", "description": "Vi planlegger å seile fra trondheim til Oslofjord uten å bruke motor, "departure_date": "Thu Feb 18 09:42:52GMT+01:002021" ”arrival_date": "Thu Feb 18 09:42:52GMT+01:002021" "latitude": "44.45", "longitude": "44.45", "positions": "LongBlob",} "tag": "getTripByToken",}

Error respons	status:401	innhold: error : “Uautorisert”

Metodenavn	**PUT updateTrip**	
url	https://strange-mind-305617.ew.r.appspot.com/trips/{token}	
Bruk	// Redigerer en spesifikk tur til en bruker	
body	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "trip_id": "1", "boat_id": "1", "tilte": "motoren har stoppet", "destination": "Oslofjord", "registered #0, #1": liste til medreisende som blir lagt med på en tur og som HAR konto i systemet "unregistered #0, #1": liste til medreisende som blir lagt med på en tur og IKKE har konto i systemtet "remove #0, #1": liste til medreisende som blir fjernet fra en tur }	
Vellykket	Status:200	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI

Respons		0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "updated": "boolean" "tag": "updateTrip",}
Error respons	status:401	innhold: error : “Uautorisert”

Metodenavn	**PUT setArrivalDate**
url	https://strange-mind-305617.ew.r.appspot.com/trips/arrival
Bruk	// Avslutt en tur og set ankomstdato til turen enten automatisk ved å hente nåtid eller egendefinert fra brukeren
body	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "trip_id": "1", "boat_id": "1", "arrival_pos": "LongText", ”arrival_date": "Thu Feb 18 09:42:52GMT+01:002021"}

Vellykket **Respons**	Status:200	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "updated": "boolean" "tag": "setArrivalDate",}
Error respons	status:401	innhold: error : “Uautorisert”
	status:400	innhold: error : “Feil ved anmodning”

Metodenavn	**PUT setDepartureDate**
url	https://strange-mind-305617.ew.r.appspot.com/trips/departure
Bruk	// Redigerer både ankomstdato og sted
body	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "trip_id": "1", "boat_id": "1",

	"departure_pos": "Longtext", ”departure_date": "Thu Feb 25 09:42:52GMT+01:002021"}	
Vellykket **Respons**	Status:200	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "updated": "boolean" "tag": "setDepartureDatge",}
Error respons	status:401	innhold: error : “Uautorisert”
	status:400	innhold: error : “Feil ved anmodning”

Metodenavn	**PUT updatePostions()**
url	https://strange-mind-305617.ew.r.appspot.com/trips/positions
Bruk	// Det endepunktet brukes til å spore posisjonen i sanntid og lagre det i databaen
body	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ

	4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "Position": "Longtext", "boat_id": "1", "departure_pos": "Longtext",}	
Vellykket **Respons**	Status:200	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "updated": "boolean" "tag": "updatePostions",}
Error respons	status:401	innhold: error : "Uautorisert"
	status:400	innhold: error : "Feil ved anmodning"

Metodenavn	**PUT updatePermission()**
url	https://strange-mind-305617.ew.r.appspot.com/trips/user_permission
Bruk	// Endrer rettigheten til en bruker lagt til turen som medreisende. 0 har kan redigere på andre sin logginnlegg (admin)

	1 har lese og skriverettigheter (medreiser) 2 har kun lese rettigheter (turen har blitt delt med hen)	
body	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "permission": "0", "trip_id": "1", "user_id": "1", // brukeren som permisjonen skal bli endret "departure_pos": "Longtext",}	
Vellykket **Respons**	Status:200	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "updated": "boolean" "tag": "updatePermission",}
Error respons	status:401	innhold: error : “Uautorisert”
	status:400	innhold: error : “Feil ved anmodning”
	status:500	innhold: error : “Intern tjenerfeil”

Metodenavn	**PUT updateConfirmed()**	
url	https://strange-mind-305617.ew.r.appspot.com/trips/user_confirmed	
Bruk	// Sett at brukeren har akseptert eller avlyst å bli lagt på en tur som medreiser	
body	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "trip_id": "1",,}	
Vellykket **Respons**	Status:200	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "updated": "boolean" "trip_id": "1" "tag": "updateConfirmed",}
Error respons	status:401	innhold: error : “Uautorisert”
	status:400	innhold: error : “Feil ved anmodning”

	status:500	innhold: error : “Intern tjenerfeil”

Metodenavn	**DELETE deleteTrip()**	
url	https://strange-mind-305617.ew.r.appspot.com/trips/{trip_id}/{token}	
Bruk	// Slett en tur	
body	Tom, data som trenges blir sendt i uri som parameter	
Vellykket **Respons**	Status:200	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "tag": "deleteTrip",}
Error respons	status:401	innhold: error : “Uautorisert”
	status:400	innhold: error : “Feil ved anmodning”

6.3 Notification.java

Metodenavn	**POST postNotification()**	
url	https://strange-mind-305617.ew.r.appspot.com/notifications	
Bruk	// Registrer en et nytt varsel når brukeren blir lagt til en båt eller tur	
body	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "content": "You have been invited to boat...", "user_id": "3", // brukeren som får varsel "trip_id": "3", // enten den eller boat_id er null "boat_id": "3", "longitude": "10.23",}	
Vellykket **Respons**	Status:201	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "tag": "postNotification",,}
Error respons	status:401	innhold: error : “Uautorisert”

	status:400	innhold: error : “Feil ved anmodning”
	status:500	innhold: error : “Intern tjenerfeil”

Metodenavn	**GET getNotificationByUserId()**	
url	https://strange-mind-305617.ew.r.appspot.com/notifications/{token}	
Bruk	// Henter alle varsler brukeren får og puller hvert et halvt minutt	
body	Tom, data som trenges blir sendt som parameter i uri	
Vellykket **Respons**	Status:200	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "tag": "getNotificationByUserId",}
Error respons	status:401	innhold: error : “Uautorisert”

Metodenavn	**PUT markAsSeen()**

url	https://strange-mind-305617.ew.r.appspot.com/notifications/seen	
Bruk	// Setter at et varsel har blitt sett og åpnet av brukeren	
body	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "notification_id": "1",}	
Vellykket **Respons**	Status:200	{ "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI0IiwiaWF0IjoxNjIwNzQ4ODU4LCJleHAiOjE2MjA3NDk3NTh9.spCc4MIzBd9_ROxOjneB6aIcK1aqyc3Qpg95mGGnvAM "updated": "boolean: true" "tag": "markAsSeen",}
Error response	status: 400	innhold: error : “Feil ved anmodning”
	status:401	innhold: error : “Uautorisert”
	status:401	innhold: error : “Ikke funnet”

7. Sikkerhet

7.1 Kryptering og digitale sertifikater

Systemet vårt krypterer data som sendes mellom klient og server. Dette gjøres ved hjelp av HTTPS, som er en sikrere utgave av HTTP-protokollen. HTTPS benytter seg av enten TLS (*Transport Layer Security*) eller SSL (*Secure Sockets Layer*) for å kryptere data. I prosjektet vårt bruker vi en Google Cloud-tjenesten, og den benytter seg av SSL. Google Cloud bruker SSL-sertifikater for kommunikasjon mellom klienten og belastningsfordeler. En belastningsfordeler er en enhet som distribuerer nettverkstrafikk på flere ulike ressurser. I Google Cloud fungerer det slik at belastningsfordeleren har et SSL-sertifikat og dette sertifikatets korresponderende private nøkkel. Forbindelsen opprettes via et såkalt SSL-håndtrykk, hvor klienten forespør en sikker forbindelse, serveren viser SSL-sertifikatet sitt, og klienten validerer det.

For å kunne logge inn brukere uten nett var vi nødt til å lagre passordet et sted på klienten, siden vi ikke kan sjekke data som ligger i databasen. Løsningen vi valgte var å lagre passordet lokalt i "SharedPreferences". Vi ville unngå å lagre passordet som ren tekst selv om det blir lagret i privat modus med en nøkkel som trengs for å få til det, og det er kun appen som har denne tilgangen. Dette har vi unngått ved å kryptere passordet før det blir lagret ved bruk av en kryptonøkkel kalt MasterKey og KeyGenParmeterSpec.

7.2 Hashing og salting

I løsningen vår har vi gjort en del sikkerhetstiltak. For det første har vi tatt i bruk hash og salt ved lagring av en brukers passord, slik at vi sikkert får lagret brukerens passord i databasen. For å få til dette bruker vi SecureRandom til salting og KeySpec, som er et Java-bibliotek for kryptografi, til hashing.

```java
public byte[] generateSalt(){
    SecureRandom random = new SecureRandom();
    byte[] salt = new byte[16];
    random.nextBytes(salt);
    return salt;
}

public byte[] generateHash(String password, byte[] salt){
    /**
     * PBEKeySpec(password.toCharArray(), salt, iterationCount, keyLength)
    // password - the password.
    // salt - the salt.
    // iterationCount - the iteration count.
    // keyLength - the to-be-derived key length.
     */
    KeySpec spec = new PBEKeySpec(password.toCharArray(), salt, 1000, 128);
    try {
        SecretKeyFactory factory = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA1");
        byte[] hash = factory.generateSecret(spec).getEncoded();
        return hash;
    } catch (NoSuchAlgorithmException e) {
        throw new IllegalStateException("Missing algorithm: PBKDF2WithHmacSHA1", e);
    } catch (InvalidKeySpecException e) {
        throw new IllegalStateException("Invalid SecretKeyFactory", e);
    }
}
```

Figur 7.1 skjermbilde som viser hash og salt funksjon blir brukt i prosjektet

I figuren (7.1) over ser vi at funksjonen generateSalt returnerer et tilfeldig generert array bestående av 16 bytes. Salt brukes sammen med generering av hash for å forhindre angrep ved bruk av regnbuebord. Regnbuebord er store tabeller fylt med hash-verdier som brukes til å finne passord til en hash verdig ved å snu hashing-funksjonen. Ved å bruke et unikt salt til hvert passord, vil hver resulterende hash-verdi bli unik, og dermed vil angrep med regnbuebord ikke være effektive. Dette oppnår vi ved å bruke SecureRandom som er nevnt tidligere i dette avsnittet.

Funksjonen generateHash tar i mot passordet og konverterer det til et char-array. Dette er logisk da objekter fra String-klassen er immutable, og i vårt tilfelle vil vi overskrive det gamle passordet når vi er ferdige med å generere hash. Den tredje parameteren i PBKeySpec er iterasjonstallet, som angir antall ganger passordet hashes under avledningen av den symmetriske nøkkelen. Jo høyere iterasjonstallet er, desto vanskeligere er det å validere passordet, og i vårt system har vi satt tallet til å være 1000. Til slutt har vi nøkkellengde som er antall bits symmetriske nøkkelen får. Les mer om hash og PBKeySpec her.

Hash-og salt-verdiene er de som lagres i databasen. For å verifisere passordet ved f.eks. innlogging har vi en metode som sammenligner hash-verdien i databasen med den som blir generert når passordet brukeren sender inn sammen med saltet i databasen hashes. Hvis hash-verdiene er like, blir brukeren logget på, dersom de ikke er det får brukeren tilbakemelding om at enten epost eller passord er feil.

7.3 Autentisering

```
public String refreshToken(String oldToken){
    if(oldToken == "") return "";
    try{
        Jws<Claims> jws = validateToken(oldToken);
        if(jws == null) return "";
        Instant now = Instant.now();
        String newToken = Jwts.builder().setSubject(jws.getBody().getSubject()).setIssuedAt(Date.from(now))
        .setExpiration(Date.from(now.plus(EXPIRATION_TIME,ChronoUnit.MINUTES))).signWith(pKey).compact();
        return newToken;
    }
    catch(Exception e){
        e.printStackTrace();
    }
    return "";
}

public  Jws<Claims> validateToken(String jwtString) {
    try {
        String encodedKey = Base64.getEncoder().encodeToString(pKey.getEncoded());
        Key hmacKey = new SecretKeySpec(Base64.getDecoder().decode(encodedKey),
        SignatureAlgorithm.HS256.getJcaName());

        Jws<Claims> jwt = Jwts.parserBuilder()
        .setSigningKey(hmacKey)
        .build()
        .parseClaimsJws(jwtString);

        return jwt;
    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
}
```

Figur 7.2 skjermbilde over funksjoner som generer og validerer tokenet i prosjektet

Et annet sikkerhetstiltak som vi har hatt fokus på i systemet vårt er bruken av JSON Web Token for autentisering av brukere og hvilke rettigheter de har mens de bruker applikasjonen. Tokenet blir generert ved hjelp av en privat nøkkel som genereres ved bruk av "SignatureAlgorithm" som bruker algoritmen *HS256* som videre bruker *SHA-256* kryptografisk hashfunksjon. Denne funksjonen genererer en hashverdi av type bitmatrise

med fast størrelse. Første gang tokenet blir generert er når brukeren logger inn, og dette tokenet brukes videre for å verifisere at brukeren er den hen faktisk er. Autorisering-mekanismen i applikasjonen vår fungerer slik at i hvert kall til server sender klienten det siste tokenet den har fått fra serveren med i forespørselen for å bli validert på server-siden. Dersom tokenet er gyldig returnerer vi data ut ifra andre data som er forespørselen er avhengig av. Hvis ikke sender vi HTTP-statuskode 401 ("Uautorisert") til klienten som logger brukeren ut med tilbakemelding om at sesjonen er utløpt. Etter diskusjon og avtale med produkteieren bestemte vi oss for å sette tokenets utløpstid til 15 minutter.

Vi har også noen sikkerhetstiltak for å bidra til å sikre at brukeren er den hen utgir seg for å være. Dersom brukeren ønsker å endre brukerinfo eller bytte passord, må det bekreftes med det gamle passordet.

7.4 Beskyttelse mot SQL-injection

A1 injeksjon: For å unngå dette problemet har vi brukt PreparedStatement for SQL-spørringer. Det vil si at alt av SQL-kode er definert først, og deretter sender vi hver parameter til spørringen, slik at databasen vår skiller mellom kode og data uavhengig av det som klienten skriver. I ett tilfelle hvor brukeren skriver "SELECT * FROM accounts WHERE accountID/name = 1 OR 1=1;", vil spørringen se etter ID/name som er bokstavelig likt hele strengen.

7.5 Beskyttelse mot cross-site scripting (XSS)

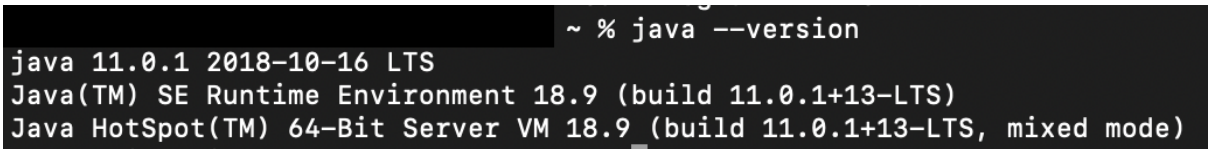
A3 cross-site scripting (XSS): Vi genererer ingen XML basert på brukeren input. Brukerinput behandles som rene strenger. Resultater ut ifra brukerinput er testet på samme måte som funnet på denne nettsiden. Ingen av inputfeltene våre har egen metode-attributt da vi håndterer dem i egne funksjoner.

8. Installasjon og kjøring

8.1 Krav

For å kunne kjøre applikasjonen må du sikre at du har JDK Java, Maven, Android Studio og git installert. Dette kan du gjøre ved å følge bruksanvisning under:

1. **Installer Java (JDK)**:



Figur 8.1 skjermbilde over JDK som brukes i prosjektet

   a. For Windows og Mac: Alle teammedlemmer har brukt Oracle sin nettside til å installere Java. Vi bruker versjon 11 som vises i figuren (8.1)
      i. https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html
2. **Installer git**: Det er flere måter å laste ned Git på, men det som er lettest å installere den direkte fra linken: Downloads, Dersom du foretrekker å gjøre det gjennom terminalen kan du følge punktene under.
   a. For Mac:
      i. Ved å kjøre kommandoen git "git --version" i terminalen. Hvis du ikke allerede har den installert, vil den be deg om å installere den.
   b. For Linux:
      i. Du kan kjøre kommandoen " sudo dnf install git-all" eller "sudo apt install git-all" hvis du bruker debian-basert distribusjon f.eks (Ubuntu)
   c. For Windows:
      i. Det er flere måter å laste Git på på Windows, men det som er lettest å installere den direkte fra linken i punkt 1 fordi det kreves litt lang kommando i windows.
3. **Installer Maven:**
   a. For Windows
      i. Ett teammedlem har installert Maven ved å følge instruksjoner som står her, og mente at det var lett å følge etter instruksjoner.

ii. Et annet alternativ er å følge instruksjoner på denne siden, som inneholder instruksjoner for å installere både JDK og Maven.

b. For Mac:

i. Dersom du bruker OS X, kommer maven bygd inn ved å kjøre java vil maven bli automatisk installert.

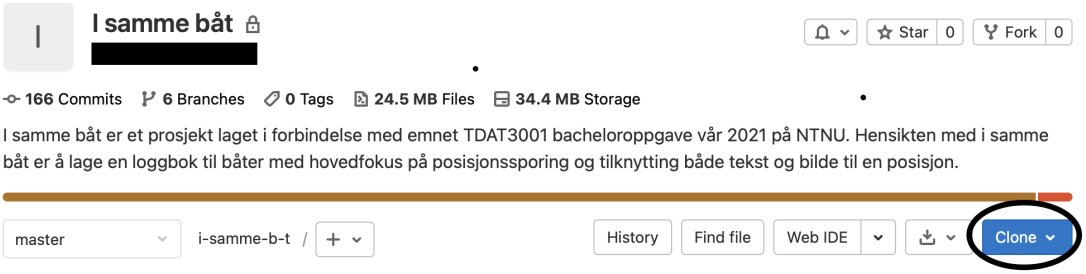
ii. Et annet alternativ er å installere den fra Apache Maven , eller å kjøre: *brew install maven*

4. **Installer android studio:**
   a. For både Mac og Windows har teammedlemmene brukt Android sin offisielle nettside til å laste ned android studio.
   b. I tillegg til android studio har vi brukt Visual Studio code til å jobbe med server og den kan installeres via denne linken

8.2 Klone prosjektet

For å klone prosjektet kan man gjøre det enten via HTTPS eller SSH nøkkel.Her er følgende instruksjoner som man kan følge for å klone prosjektet

1. Gå til hovedside til prosjektet i Git
2. Ved å trykke på clone som vist i figuren (8.2) får man to alternativer å kloneprosjektet på
   a. HTTPS: "https://gitlab.stud.idi.ntnu.no/[userName]/i-samme-b-t.git" Teammedlemmene har kun brukt HTTPS-nøkkel for å klone prosjektet
   b. SSH: "git@gitlab.stud.idi.ntnu.no:userName/i-samme-b-t.git "



Figur 8.2 skjermbilde som navigerer clone-prosjekt knappen i Git

3. Kjør i terminalen: git clone *https://gitlab.stud.idi.ntnu.no/[userName]/i-samme-b-t.git*

8.3 Konfigurasjon og bygge

8.3.1 Server

Først kan man definere følgende variablene i filen application.properties som ligger i pathen: *server/getting-started/src/main/resources/application.properties*

- **For utvikling**
    - quarkus.package.type=uber-jar
    - quarkus.datasource.jdbc.url=jdbc:mysql:///i_samme_baat?[cloud uri]
    - quarkus.datasource.db-kind=mysql
    - quarkus.http.limits.max-body-size=1000000
    - quarkus.http.limits.max-header-size=1000000
    - quarkus.http.limits.max-chunk-size=1000000
    - %dev.quarkus.datasource.username=[username]
    - %dev.quarkus.datasource.password=[password]
    - %dev.quarkus.datasource.jdbc.url=jdbc:mysql:///i_samme_baat?[local mysql uri]

Videre kan man kjøre følgende kommandoer fra rotmappen av prosjektet. Teammedlemmene har brukt som sagt to forskjellige IDEer til server brukte vi Visual Studio Code, og til klientside brukte vi Android Studio.

1. Open server mappe i Visual Studio Code eller en annen IDE du velger
2. cd getting-started
3. open terminal
4. *./mvnw compile quarkus:dev* - Kjører server
5. *./mvnw clean test* - kjører alle teste som ligger i test mappa
6. *./mvnw jacoco:report* - Finner kodedekning av alle tester
7. *./mvnw clean package* - Rydder opp target mappa, og generer en ny jar-file som resultat. Jar filen i vår prosjektet brukes til å deploye serveren til google cloud, og dette gjøres ved å kjøre kommando i neste punktet
8. *gcloud app deploy target/getting-started-1.0.0-SNAPSHOT-runner.jar*

8.3.2 Klient

Man trenger ikke å gjøre så mye i klientside for å bygge og kjøre applikasjonen. Ved å åpne *myapp* mappa i Android studio kommer Android studio mest sannsynlig til å synkronisere

prosjektet og bygge med Gradle-filene. Hvis det ikke skjer automatisk kan man gjøre det via å gjøre følgende

1. Open Android studio
2. Naviger file på toolbaren øverst og trykk på
3. Trykk på “ Sync project with Gradle files”

Etter at synkronisering er ferdig kan man enten kjøre applikasjonen i emulator eller i egen mobile ved å koble mobilen til PCen og trykke på kjøre knappen i Android studio

9. Dokumentasjon av kildekode

Både produkteieren og veilederen mente at de ikke fokuserer på dette punktet. Derfor har temaet blitt enig med dem om å fokusere på arbeidet mot problemstillingen enn å bruke tid på dokumentasjon av kildekode.

10. Kontinuerlig integrasjon og testing

10.1 Testing

10.1.1 Server-testing

Vi begynte med testing av produktet vårt så tidlig vi kunne særlig med server-testing som vi startet med i sprint 2. I starten hadde vi satt testene ved brukt av JEST og DAO for testing av back-end, hver tabell i databasen (bortsett fra join-tabeller, som er integrert inn i andre filer) har sin egen DAO-fil med et interface for tabellen, og ulike funksjoner som kjører SQL-spørringer for tabellen. For hver DAO-fil har vi satt en tilhørende testfil hvor vi tester funksjonene i DAO-filene.Noen eksempler på typer spørringer vi brukte først var som følger: finn alle tupler i en tabell, finn en tuppel som har en bestemt ID, sett inn en ny tuppel, oppdater en ny tuppel, slett en tuppel. En type spørring som også går igjen er å finne tupler basert på fremmednøkler i tabellen.

Dette måtte vi forandre på etter ønske fra produkteieren som foretrekket intigrasjon-testing over enhetstesting. Dette innebærer å teste endepunktene og responsen fra serveren i stedet i for databasespørringer og dao-klassene. Derfor byttet vi hele oppsettet til server-testing. For å få dette til å fungere la vi til noen plugins i *pom.xml* i path server/getting-started, som vises nederst i dette avsnittet. Til det nye oppsettet brukte vi quarkus-bibliotek for testing med Junit5 for å få til å teste integrasjonstesting av HTTP-statuskoder og H2 database type for innebygd database. H2 er en innebygd database. Den kan kjøre som en server, basert på en fil, eller leve helt i minnet. Dette brukte vi nemlig til kjøre import.sql som inneholder SQL-scriptet med testdata til databasen. Ved bruk av H2 fikk vi til å kjøre filen før hver testfunksjon på innbygd database (in memory database). Som nevnt tidligere har vi satt opp testing på nytt etter ønske fra produkteiere, som hadde et ønske også å dekke 90% eller mer av koden i testing. Derfor har vi testet mest mulig hver eneste sjekk og responsen i serveren for hvert eneste endepunkt.

For å måle hvor mange linjer i koden vår blir utført under automatiske testene vi har i serveren brukte vi jacoco som er nevnt og forklart i avsnitt 3. For å få til å teste mest mulig applikasjonen som helhet og ikke bare enkelte endepunkter har vi skrevet testene slik at flere enn ett endepunkt blir kalt i en testfunksjon. Et eksempel på dette er som følger: registrer og hent data, update og hent eller slett og hent som vil returnere statuskode “Ikke funnet”.

Figuren (10.1.1.1) under er et eksempel på testene som kaller flere endepunkter i samme testfunksjon. Ulempen med dette rammeverket er at vi ikke kan se kodedekning dersom én av testene feiler

For konfigurasjon av testing defineres de følgende variablene i filen application.properties som ligger i pathen: *server/getting-started/src/main/resources/application.properties*

- **Variabler**
    - quarkus.package.type=uber-jar
    - quarkus.datasource.db-kind=h2
    - dev.quarkus.datasource.username=[username]
    - dev.quarkus.datasource.password=[password]
    - quarkus.http.limits.max-body-size=1000000
    - quarkus.http.limits.max-header-size=1000000
    - quarkus.http.limits.max-chunk-size=1000000
    - quarkus.datasource.jdbc.driver=org.h2.Driver
    - quarkus.datasource.jdbc.url=jdbc:h2:tcp:///localhost/mem:test;MODE=MySQL;INIT=RUNSCRIPT FROM ‘src/test/resources/import.sql’

- Kommandoer
    - cd server/getting-started
    - *./mvnw clean test*
    - Da alle testene er kjørt og var vellykket, kan man kjøre kommandoen *./mvnw jacoco:report*
        - Den kommandoen generer en *index.html* fil i server/getting-started/target/site/index.html

```java
@Test
Run Test | Debug Test
public void testUpdatePortNoToken() {
    String token = login("c@c.c", "Cre123");
    Map<String,String> data = new HashMap<String,String>();
    data.put("token", token);
    data.put("name", "The house of the boat");
    data.put("latitude", "11.22");
    data.put("longitude", "102.10");
    byte[] b = convertMap(data);

    given().body(b)
    .when().post("/ports")
    .then()
    .statusCode(201);

    Map<String,String> data1 = new HashMap<String,String>();
    data1.put("token", "no token");
    data1.put("name", "The house of the boat");
    data1.put("latitude", "11.22");
    data1.put("longitude", "102.10");
    data1.put("port_id", "1");

    given()
    .body(convertMap(data1))
    .when().put("/ports")
    .then()
    .statusCode(service.TOKEN_EXPIRATION_STATUS)
    .extract();
}
```

Figur 10.1 skjermbilde viser en testfunksjon som kaller flere endepunkter

```xml
<dependency>
    <groupId>io.quarkus</groupId>
    <artifactId>quarkus-junit5</artifactId>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>io.rest-assured</groupId>
    <artifactId>rest-assured</artifactId>
    <scope>test</scope>
</dependency>
```

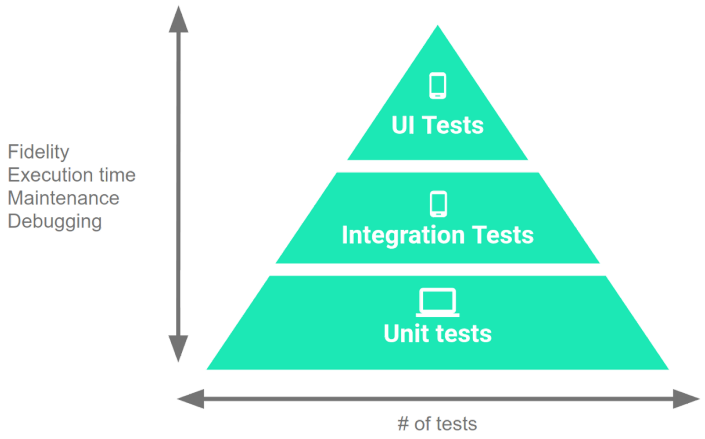
```xml
<plugin>
    <artifactId>maven-surefire-plugin</artifactId>
    <version>${surefire-plugin.version}</version>
    <configuration>
       <systemPropertyVariables>

<java.util.logging.manager>org.jboss.logmanager.LogManager</java.u
til.logging.manager>
          <maven.home>${maven.home}</maven.home>
       </systemPropertyVariables>
    </configuration>
</plugin>
```

Figur 10.2 skjermbilde over avhengigheter for quarkus-test

Figur 10.3 skjermbilde over avhengigheter for quarkus-test

10.1.2 Klient-testing



Figur 10.4 viser forskjellige mulige klienttesting Fundamentals of Testing

Det finnes mange måter å teste klientsiden på. Alt fra små, UI tester som tester enkelte komponenter (også kalt enhetstester), til integrasjonstester som tester sammenhengen mellom moduler til store ende-til-ende tester som tester sammenhengen i tilnærmet hele systemet. Produkteier ønsket at vi fokuserte på integrasjonstester så vi endte opp med å skrive en blanding av integrasjons og enhetstester. Vi fokuserte også på å kjøre testene på virkelige enheter eller emulatorer over å simulere det på tilegnede programmer. Dette betyr i praksis at vi vil få mer nøyaktige resultater som reflekterer hva som ville skjedd på ekte apparater, men også at testene bruker mye lengre tid på å kjøre.

For å sette opp testene brukte vi JUnit 4 og espresso som rammeverk. JUnit gir oss tilgang til metoder som forteller koden rekkefølgen av kjøring. “@Before” og “@After” er for eksempel greie hjelpestikkord som forteller at en metode skal kjøre før eller etter hver enkelte test. Testmetodene blir i tillegg markert med “@Test”. “@Rule” kommer også fra JUnit og den sørger for å sette opp valgt aktivitet før hver test kjøres og å lukke den etter endt test. Espresso brukes for å utføre de ulike testene vi kjører inni testmetodene. For eksempel kan det være å sjekke om to verdier er like eller om konteksten har en spesifikk intent. Oppsettet på espressotester er som følger:

- onView(), velger hvilken komponent som det skal gjøres noe med. Som parameter tar den inn blant annet id-en til komponenten eller et utdrag av teksten som komponenten skal inneholde.
- perform(), handlingen som skal utføres på komponenten. Som argument kan den ta inn blant annet at den skal klikke på komponenten eller skrive inn en spesifisert tekst.

- check(), sjekken vi gjør for å se om den ønskede hendelsen ble utført. Den kan ta inn blant annet et objekt av typen matcher som utfører en sammenligning med spesifiserte forhold (f. eks. om komponenten inneholder en satt tekst) eller sjekke om komponenten er synlig.

Disse metodene kan brukes i forskjellig oppbyggelse og trenger ikke alltid inneholde en perform() eller en check() om man ikke skal utføre en handling eller sjekk på den samme komponenten.

For å utføre grundige tester som kan teste at klienten fungerer som den skal må vi ha en måte å få svar fra serveren. Når vi kjører tester vil vi ikke at den virkelige serveren skal bli brukt og påvirke databasen vår, derfor bruker vi WireMock. Den fungerer slik at vi lar kallene til serveren gå gjennom en opptaker som lagrer alle kall til serveren og alle svar vi får tilbake. Det betyr at vi kan kjøre klienten en gang og gå innom alle funksjonalitetene vi vil bruke i testene og lagre de svarene vi får fra serveren til å bruke i testene. På den måten vil vi alltid få samme svar og ikke være avhengig av en faktisk tilkobling til serveren. Ved kjøring av testene setter vi opp en tilknytning til en lokal kjøring av WireMock som får inn en spørring fra klienten og ser om den har en lagret respons som passer til spørringen.

Det er ikke alltid WireMock fungerer perfekt til vår bruk. Første hindring er at vi ofte sender ved token som et parameter i url-en til serveren. Siden den er forskjellig i hvert eneste kall, vil WireMock aldri kunne finne en respons som passer med spørringen. Vi løste dette på to ulike måter. Ved kjøringen av klienten for å gjøre opptak av http-kall endret vi serveren til å ikke endre tokenet, men heller sende tilbake det samme som den fikk inn. Det gjør sjekkingen mye enklere for WireMock. I tillegg endret vi manuelt noe av den lagrede dataen WireMock bruker til å isteden for å se etter et eksakt treff på url-en til å se om den inneholder deler av url-en. På den måten kan vi fjerne tokenet fra likningen.

En annen faktor med WireMock som måtte rettes ut var om vi skulle kjøre to spørringer til server og forvente to ulike svar tilbake. Dette skal i utgangspunktet ikke være et problem da WireMock har et system som heter scenario. Om en spørring blir lagret to ganger vil WireMock gi dem et scenario hver. Den første vil bli lagret som den første spørringen og samtidig vil den sette scenarioet til neste gang vi får samme spørring til å gå til neste respons den har lagret. Et problem med dette er at vi har mange spørringer som kjøres konstant i bakgrunnen av applikasjonen som poller etter data og da vil WireMock tilslutt gå tom for

responser å hente fra og krasjer som en respons. Et annet problem er om vi vil kjøre tester som bruker de samme kallene flere ganger har ikke WireMock kontroll på når en test starter og slutter og dermed vet den ikke når den skal tilbakestille seg til å bruke den første responsen igjen. For å løse disse problemene lagde vi manuelt responser som sjekket deler av url-en og som ikke brukte scenarioer for de gjennomgående spørringene våre og mellom hver test kjører vi et http-kall som er til for å tilbakestille WireMock.

10.2 Kontinuerlig integrasjon

```yaml
image: maven:3-jdk-11

stages:
  - build
  - test-server
# Set up MySQL test-database
services:
  - mysql:5.7

# Configure test-database
variables:
  MYSQL_DATABASE: [redacted]
  MYSQL_ROOT_PASSWORD: [redacted]
  NODE_ENV: ci

build:
  stage: build
  script:
    - cd server
    - ls
    - cd getting-started
    - mvn install
  only:
    - master
```

```yaml
# Run JEST tests
run_tests_server:
  stage: test-server
  script:
    - cd server
    - ls
    - cd getting-started
    - ls
    - mvn clean test
    - ls
  only:
    - master
```

Figur 10.5 skjermbilde viser oppsettet til kontinuerlig test og faser blir brukt

Figur 10.6 skjermbilde viser server test-fase i kontinuerlig test

Til kontinuerlig integrasjon testing bruker vi CI/CD i Gitlab. Måten vi implementerte dette på var å sette opp pipeline i prosjektet mot Gitlab slik at vi kjører testene hver gang vi pusher til master branchen. I figuren (10.2.1) over ser vi et skjermbilde av filen *.gitlab-ci.yml* i server-mappa. Helt øverst i fila lager vi et docker-bilde til å kjøre maven-prosjekt som backend i prosjektet vårt er bygd på. I denne filen lager vi også en virtuell database med brukernavn og passord under *service* fase. Vi har også satt opp to faser den første er "build" til å installer maven og bygge en virtuell server. Dersom første fasen lykkes starter den andre fasen "test-server" og pipeline feiler eller lykkes avhengig av selve testene. I begge fasene i figurene under dette avsnittet ser vi at vi har med only: master som betyr at testene kjøres kun

i master som er beskyttet branch og vi brukte å oppdatere den kun ved sprint-avslutning da vi var sikre på at det er ingen som feiler.

Kravdokumentasjon – I samme båt

Anvendt Informasjonsteknologi, IDI, NTNU

Bacheloroppgave vår 2021

Revisjonshistorie

Dato	Versjon	Beskrivelse	Forfatter
19/01/2021	1.0	Skrive ferdig mesteparten av førsteutkast	Mahmoud Ibrahim, Eirik Plahte og Christian Riksvold

Innholdsfortegnelse

1. Introduksjon

Dette dokumentet er skrevet i forbindelse med bacheloroppgave “i samme båt”, som er laget for å oppfylle kravene til emnet bacheloroppgave dataingeniør TDAT3001 i studieprogrammet: Bachelor i ingeniørfag, data.

Dokumentet inneholder kravdokumentasjon til systemet “i samme båt”. Dette blir brukt for kommunikasjon mellom studentene og oppdragsgiver.

2. User Stories og scenarioer

Som	bruker
vil jeg	registrere bruker
slik at	jeg kan lagre mine data i systemet og bruke dem på nytt
akseptansekriterier	- Få en verifiseringsmail - Få beskjed om mailen er registrert fra før - Validere inndata og gi tilbakemelding - Passordet blir hashet og saltet - Bli sendt til innloggingsside

Som	bruker
vil jeg	logge inn
slik at	jeg får tilgang til mine data
akseptansekriterier	- Bli sendt til forside eller pågående tur - Slippe å manuelt logge inn om bruker har logget inn fra før - Validere inndata og gi tilbakemelding

Som	bruker
vil jeg	spore en tur med gps
slik at	jeg kan se hvor jeg er og hvor jeg har vært underveis på en tur
akseptansekriterier	- Se en rute på kart - Ruta skal ha mange nok punkter til å gjøre visningen naturlig - Se båtens posisjon på kartet - Kartet skal følge båten i sanntid om bruker ikke har beveget på kartet - Kunne trykke på en knapp for å følge nåværende posisjon

Som	bruker
vil jeg	se en tur på sjøkart
slik at	jeg enklere kan se detaljer på havet
akseptansekriterier	- Kartet skal være et sjøkart - Kartet skal minst dekke norskekysten

Som	bruker
vil jeg	lagre reiser med posisjonssporing
slik at	jeg i ettertid kan se hele reisen på et kart
akseptansekriterier	- Ruten blir lagret når turen er ferdig - Se ruten båten har tatt på et kart i ettertid - Kunne finne igjen ruten fra en liste med turer

Som	bruker
vil jeg	registrere båt
slik at	jeg kan holde data på forskjellige båter separat
akseptansekriterier	- Kunne legge inn informasjon om båten - Båten blir lagret i en liste over båter som brukeren er tilknyttet - Kunne legge til flere eiere av båten - Mulighet til å sette aktiv båt om bruker har flere båter fra før - Validere inndata og gi tilbakemelding - Gi tilbakemelding om båten ble registrert

Som	bruker
vil jeg	fylle inn overordnet informasjon om turen
slik at	jeg og andre brukere kan raskt få oversikt over turen
akseptansek	- Kunne lagre informasjon som gjelder for hele turen

riterier	- Knytte personer til turen - Se informasjonen i den tilknyttede turen - Validere inndata og gi tilbakemelding

Som	bruker
vil jeg	knytte tekst og bilde til en posisjon
slik at	jeg og andre brukere kan se hvor et logginnlegg hendte
akseptansek riterier	- Logginnlegget vises i ruten på kartet - Kunne legge til bilder i logginnlegget - Validere inndata og gi tilbakemelding

Som	bruker
vil jeg	se alle logginnlegg som hører til en tur
slik at	jeg kan få oversikt over alle innleggene som ble gjort på en tur
akseptansek riterier	- Alle logginnleggene til en tur vises på kartet - Alle logginnleggene til en tur vises i en liste - Kunne klikke inn på logginnlegget fra kartet - Logginnleggene i lista blir sortert kronologisk

Som	bruker
vil jeg	se tidligere turer ut fra en liste
slik at	jeg lett kan få oversikt over tidligere turer
akseptansek riterier	- Turene skal ligge sortert under sine respektive båter - Turene skal være sortert kronologisk - Alle turer blir vist på forsiden - Logginnlegg skal vises under sin korresponderende tur - Brukeren kan gå inn på turen og se mer detaljer

Som	båteier
vil jeg	se oversikt over båter
slik at	jeg kan se all informasjon om båtene mine på ett sted
akseptansekriterier	- Alle funksjonene ved båt skal være tilgjengelige fra samme side - Kan endre hvilken båt som er aktiv

Som	båteier
vil jeg	legge ved brukere og navn som medreisende i en tur
slik at	de som er med på turen kan skrive logginnlegg og har tilgang til turen
akseptansekriterier	- Få opp liste over tilknyttede brukere på båten som enkelt skal kunne bli lagt til med en avkrysningsboks - Gjester som har vært på båten før skal komme som forslag ved innskriving - Kan legge ved tidligere gjester ved navn - Brukere får varsel at de er lagt til - Brukere som blir lagt til på turen kan godta eller avslå - Båteieren kan fjerne medreisende som ble lagt med på turen

Som	båteier
vil jeg	dele en tur med en spesifikk bruker
slik at	jeg kan vise turen til en bruker som ikke var med om bord
akseptansekriterier	- Brukeren får et varsel om at hen ble lagt til - Brukeren får se en visning av turen - Kan skrive inn navnet på tidligere gjester på båten - Tidligere gjester skal dukke opp som forslag ved innskriving

Som	bruker
vil jeg	kunne få hjelp om man ikke forstår appens oppsett

slik at	jeg har et hjelpeverktøy hvis jeg ikke vet hvordan en funksjon fungerer i appen
akseptansekriterier	- Hjelpesiden skal inneholde gode forklaringer på konsepter i appen som kan være forvirrende - Hjelpesiden skal inneholde svar på ofte stilte spørsmål - Brukere får hjelp der det trengs uten å gå til hjelpesiden

Som	bruker
vil jeg	se andre båters turer
slik at	jeg kan se turer jeg selv ikke var med på
akseptansekriterier	- Se turen i sanntid - Se turen i ettertid - Flere av funksjonene for passasjerer skal være sperret

Som	bruker
vil jeg	se farten til båten i sanntid
slik at	jeg vet hvor fort jeg kjører
akseptansekriterier	- Farten blir vist på samme side som kartet (oversikten av turen) - Måleenhet er knop - Farten blir oppdatert hvert sekund

Som	bruker
vil jeg	se distansen til turen underveis
slik at	jeg vet hvor langt jeg har reist
akseptansekriterier	- Seilt distanse blir vist på samme side som kartet (oversikten av turen) - Den oppdateres når vi får en ny posisjon på kartet - Måleenhet er nautiske mil

Som	bruker
vil jeg	skrive logginnlegg til en tidligere passert posisjon
slik at	jeg kan skrive et innlegg i ettertid om vi har reist forbi posisjonen det skjedde
akseptansekriterier	- Brukeren kan trykke på en hvilken som helst tidligere posisjon på kartet - Brukeren blir sendt til siden for å legge til logginnlegg - Tidspunktet til logginnlegget blir til det lagrede tidspunktet på den valgte posisjonen

Som	bruker
vil jeg	skrive logginnlegg til et tidligere tidspunkt
slik at	jeg kan skrive et innlegg i ettertid om jeg ikke fikk gjort det da hendelsen akkurat skjedde
akseptansekriterier	- Brukeren kan velge et hvilket som helst tidspunkt - Tidspunktet blir validert - Posisjonen til logginnlegget blir satt til den nærmeste lagrede posisjonen på det valgte tidspunktet

Som	bruker
vil jeg	få hjelp til å komme i gang
slik at	jeg blir litt veiledet første gang jeg bruker appen
akseptansekriterier	- Få spørsmål og enkel link til å registrere båt - Fjernes etter brukeren har kommet i gang

Som	bruker
vil jeg	redigere informasjon om tur
slik at	jeg kan endre tittel og beskrivelse av turen underveis
akseptansekr	- Bare den som startet turen skal kunne redigere

iterier	- Kunne lagre informasjon som gjelder for hele turen - Knytte personer til turen - Brukere som er lagt til i tur blir automatisk lagt til på båten - Se informasjonen i den tilknyttede turen - Validere inndata og gi tilbakemelding

Som	bruker
vil jeg	starte sporing av tur etter at selve turen har startet, og likevel få med hele ruta
slik at	jeg kan fylle inn avgangssted og tid etter at turen er startet om jeg ikke fikk gjort det ved starten av turen
akseptansekriterier	- Brukeren kan velge startposisjon på et kart - Denne posisjonen blir lagt til i ruta etterpå - Kan velge å overskrive den nåværende startposisjonen - Valider inndata og gi tilbakemelding

Som	bruker
vil jeg	manuelt bestemme endeposisjon og endetidspunkt for en tur
slik at	jeg kan bestemme at turen stoppet på et tidligere tidspunkt og sted

akseptansekriterier	- Brukeren kan velge endeposisjon ut fra punktene på turen
	  - Tidspunkt blir satt til det lagrede tidspunktet på posisjonen
	- Brukeren kan velge endetidspunkt
	  - Posisjon blir nærmeste utfra punktene på turen
	- Valider inndata og gi tilbakemelding
	- Posisjoner lagret etter endeposisjon blir slettet

Som	bruker
vil jeg	arkivere flere turer som en del av samme reise
slik at	jeg får en mer oversiktlig historie over alle turene mine

akseptansekriterier	- Skal være enkelt å flytte turer mellom reiser - Egen side for administrering og organisering av reiser - Valider inndata og gi tilbakemelding - Endre navn og beskrivelse på reiser - Sortere reiser kronologisk - Reiser vises på forsiden, og ved å trykke på den blir turene vist

Som	bruker
vil jeg	legge til havn på kartet manuelt
slik at	jeg kan legge til havner jeg vet jeg vil trenge uten å reise dit
akseptansekriterier	- Bruker kan velge posisjonen til havnen på et kart - Andre havner er synlig på kartet - Bruker kan fylle inn navn på havnen - Valider inndata og gi tilbakemelding

Som	bruker
vil jeg	legge til havn automatisk på avgangspunkt og ankomstpunkt etter en tur
slik at	jeg kan knytte turer som går til samme sted til samme havn1’’
akseptansekriterier	- Havnen blir registrert hvis det ikke er registrert en havn der fra før - Brukeren får muligheten til å gi den nye havnen et navn når turen er ferdig - Navnet til turen blir automatisk oppdatert til “Fra [starthavn] til [slutthavn]”

Som	bruker
vil jeg	redigere både navn og posisjon til en eksisterende havn
slik at	jeg kan fikse på eventuelle feil eller endringer på havnen
akseptansekriterier	- Bruken kan velge havnen ut fra et kart - Brukeren kan velge en ny posisjon på et kart - Brukeren kan fylle inn navn på havnen

	- Navn på turer som brukte havna blir oppdatert - Valider inndata og gi tilbakemelding

Som	bruker
vil jeg	slette en havn
slik at	jeg kan fjerne havner jeg ikke bruker mer
akseptansekriterier	- Brukeren kan velge havnen ut fra et kart - Brukeren må bekrefte at hen vil slette havnen - Navn på turer som brukte havna blir oppdatert - Havnen blir fjernet fra kartet

Som	båteier
vil jeg	se havner på kart
slik at	jeg kan få en oversikt over hvilke havner jeg har
akseptansekriterier	- Alle havnene til en båt blir vist på et kart

Som	bruker
vil jeg	tilbakestille passord om glemt
slik at	jeg kan få logget inn selv om jeg har glemt passordet mitt
akseptansekriterier	- Brukeren får en mail med en kode som brukes til verifisering - Vi sjekker om mail er registrert - Brukeren skriver inn koden og blir sendt til en rediger passord side - Valider inndata og gi tilbakemelding

Som	bruker
vil jeg	redigere logginnlegg

slik at	jeg kan endre på eventuelle feil eller endringer på innlegget
akseptansekriterier	- Er tilgjengelig overalt logginnlegg vises - Bruker kan legge til og fjerne bilder - Bruker blir sendt tilbake til siden hen kom fra - Valider inndata og gi tilbakemelding

Som	båteier
vil jeg	slette logginnlegg
slik at	jeg føler meg trygg og har kontroll over dataen min
akseptansekriterier	- Er tilgjengelig overalt logginnlegg vises - Bruker blir sendt tilbake til siden hen kom fra

Som	bruker
vil jeg	få varsel
slik at	jeg blir oppmerksom på at det har skjedd en endring jeg ikke var opphavet til
akseptansekriterier	- Varselikonet blir oppdatert når brukeren får et nytt varsel - Ikonet som tilsier at man har nye varsler blir fjernet etter at brukeren trykker på det - Varsel dukker opp for brukeren i sanntid

Som	bruker
vil jeg	bekrefte å bli lagt til på en tur eller båt
slik at	jeg har kontroll på hvilke turer og båter jeg vil at min bruker skal være tilknyttet
akseptansekriterier	- På båten eller turen vises navnet med et “ikke-akseptert” merke - Visning av turen eller båten brukeren blir lagt til på dukker ikke opp før brukeren har godtatt - Brukeren kan avslå

Som	båteier
vil jeg	redigere logginnlegg jeg ikke har skrevet om jeg er admin
slik at	jeg har full kontroll på båten min og kan redigere ting som ikke passer seg
akseptansekriterier	- Alle logginnlegg tilknyttet båter brukeren er admin på kan redigeres

Som	båteier
vil jeg	slette logginnlegg jeg ikke har skrevet om jeg er admin
slik at	jeg har full kontroll på båten min og kan ta bort irrelevante innlegg
akseptansekriterier	- Alle logginnlegg tilknyttet båter brukeren er admin på kan slettes - Brukeren må bekrefte at hen vil slette logginnlegget

Som	båteier
vil jeg	slette tur
slik at	jeg kan fjerne turer som jeg ikke vil lagre
akseptansekriterier	- Admin på båt og de som har startet turen kan slette den - Brukeren må bekrefte at hen vil slette turen

Som	båteier
vil jeg	slette båt
slik at	jeg kan fjerne den lagrede dataen til båten om jeg ikke vil lagre den lenger
akseptansekriterier	- Kun de som er admin på båt kan slette den - Brukeren må bekrefte at hen vil slette båten

Som	bruker

vil jeg	søke etter logginnlegg i nærheten
slik at	jeg kan se logginnlegg jeg har tilgang på som ligger i nærheten av der jeg befinner meg
akseptansekriterier	- Brukeren kan spesifisere en radius for hvilke logginnlegg som vises - Logginnleggene innenfor radien vises på et kart - Bare logginnleggene brukeren har tilgang på vises - Brukeren kan gå inn på tilhørende tur

Som	bruker
vil jeg	se andres profiler
slik at	jeg kan se kontaktinformasjonen til andre brukere
akseptansekriterier	- Brukerne skal kunne se informasjon om hverandre

Som	bruker
vil jeg	redigere brukerinformasjon
slik at	jeg kan rette opp i eventuelle feil eller endringer
akseptansekriterier	- Brukeren må bekrefte ved å skrive inn passordet sitt - Om mail er endret sjekker vi om den finnes i databasen fra før - Validere inndata og gi tilbakemelding

Som	bruker
vil jeg	endre passord
slik at	jeg kan føle meg trygg på at dataen min er sikker
akseptansekriterier	- Brukeren må verifisere ved å skrive inn nåværende passord - Nytt passord hashes og saltes før det lagres - Brukeren må bekrefte det nye passordet

	- Validere inndata og gi tilbakemelding

Som	båteier
vil jeg	redigere båt
slik at	jeg kan rette opp i eventuelle feil eller endringer
akseptansekriterier	- Bare admin brukere kan redigere - Validere inndata og gi tilbakemelding

Som	båteier
vil jeg	legge til tilknyttede til båt
slik at	jeg enkelt kan gi mine bekjente rettigheter på båten
akseptansekriterier	- Sende mail til brukerne som blir lagt til på båten og ikke finnes i databasen - Kan ikke legge til brukere som allerede er tilknyttet båten - Brukere som blir lagt til får varsel om dette - Validere inndata og gi tilbakemelding

Som	bruker
vil jeg	kunne tilknytte gammel informasjon om ditt navn til din bruker
slik at	jeg kan få koble sammen informasjon om meg som fantes i appen før jeg registrerte meg
akseptansekriterier	- Vi sjekker om mail er registrert i systemet, men ikke har aktiv konto - Brukeren får valget om å sammenslå kontoene - Brukeren får velge hvilke tilkoblinger hen vil opprettholde

Som	båteier
vil jeg	legge til logginnlegg som tilhører en båt
slik at	jeg kan holde styr på hendelser utenom turer

akseptansekriterier	- Kunne legge til bilder i logginnlegget - Validere inndata og gi tilbakemelding - Logginnlegget vises på forsiden

Som	bruker
vil jeg	søke gjennom tidligere turer
slik at	jeg enkelt kan finne igjen turen jeg leter etter
akseptansekriterier	- Søke basert på tittel og beskrivelse på logginnlegg på turen - Kan sortere turer med tanke på båtene de var på

Som	bruker
vil jeg	bytte mellom ulike typer kart
slik at	jeg kan få annerledes informasjon på kartet som ikke sjøkart gir
akseptansekriterier	- Kan enkelt endre mellom ulike typer kart med et knappetrykk - Kartet skal endres umiddelbart

Som	bruker
vil jeg	“like” et logginnlegg
slik at	jeg kan gi en positiv tilbakemelding til andre brukere
akseptansekriterier	- Brukeren kan like et logginnlegg ved å trykke på et like-ikon på turer som er delt med hen - Forfatteren får et varsel om at brukeren likte logginnlegget - Antall likes vises på innlegget

Som	bruker
vil jeg	slette konto
slik at	jeg kan være sikker på at data jeg ikke vil skal bli lagret blir fjernet

akseptansekriterier	- Brukeren må bekrefte før hen sletter konto - Brukeren blir logget ut - All data om brukeren blir slettet

Som	bruker
vil jeg	kommentere en logg
slik at	jeg kan uttrykke meg om andre brukeres innlegg direkte i appen
akseptansekriterier	- Brukeren kan kommentere et logginnlegg via en tekstboks under innlegget - Forfatteren får varsel om kommentaren - Kommentarene vises i kronologisk rekkefølge under logginnlegget

Som	bruker
vil jeg	logge inn med bruker fra andre sosiale medier som facebook eller google
slik at	jeg slipper å lage ny brukerinformasjon deriblant nytt passord
akseptansekriterier	- Ved registrering får brukeren valget om dette - Henter informasjon fra kontoen brukeren oppgir - Brukeren kan i ettertid endre brukerinfo uavhengig av andre sosiale medier

Som	båteier
vil jeg	administrere hvem som kan se min posisjon
slik at	jeg kan være sikker på at min posisjon ikke deles med uønskede
akseptansekriterier	- Velge fra en liste med kontakter hvem som kan se min posisjon

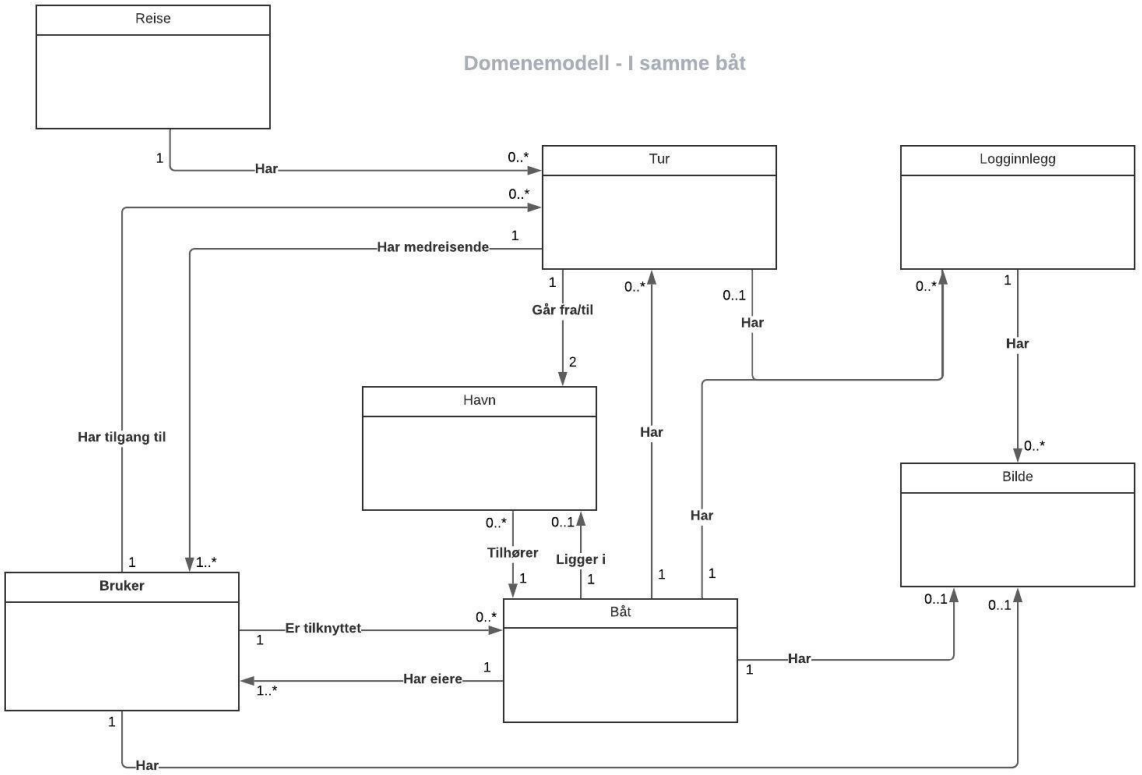
Som	båteier
vil jeg	bli varslet om brukere i nærheten
slik at	jeg kan møte andre brukere jeg kjenner når jeg er på tur i nærheten av dem

akseptansekriterier	- Brukerne må være innenfor en viss radius - Begge brukerne får varsel - Begge brukerne må ha godtatt at den andre skal se dens posisjon

Som	båteier
vil jeg	se alt fra applikasjonen på en tilknyttet nettside
slik at	jeg kan få oversikten over dataen på en større skjerm
akseptansekriterier	- Samme data skal være tilgjengelig på nettsiden som i appen

Som	båteier
vil jeg	spore forbruk (vann, olje, diesel)
slik at	jeg vet hvilken tilstand båten er i og når noe må fylles
akseptansekriterier	- Informasjon om forbruk lagres

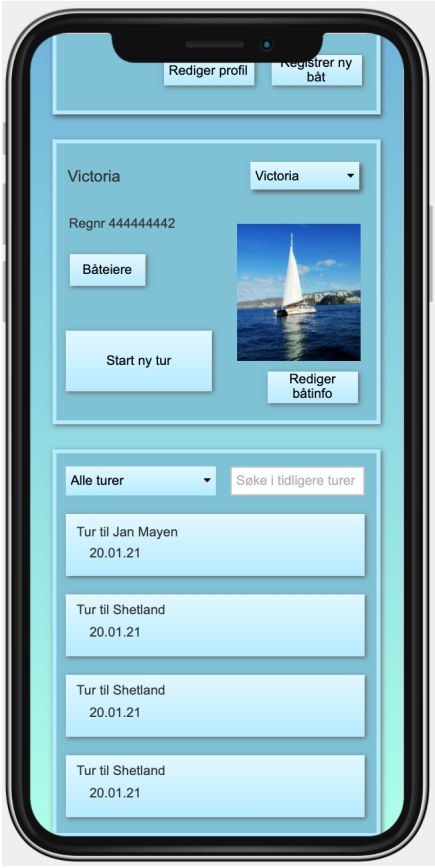
3. Domenemodell



4. Prototyper

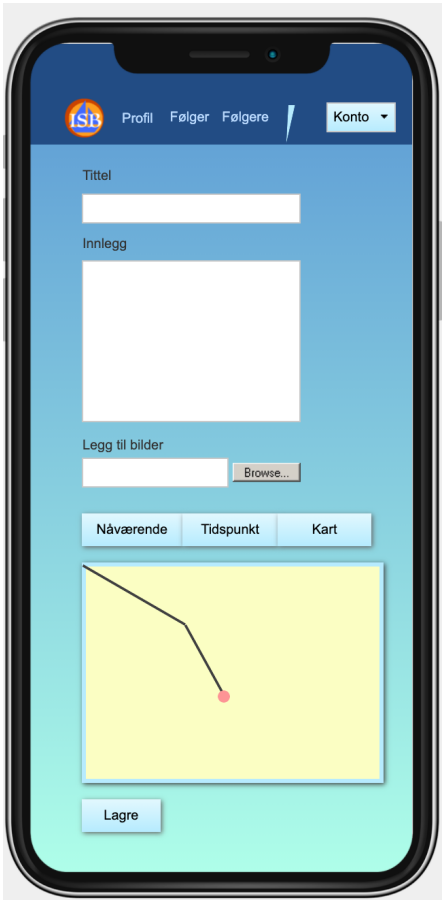
Til å kommunisere med produkteieren angående desing, app-struktur og hvordan vi tenkte å løse oppgaven på brukte vi Wireframes. Tjenesten som ble brukt var Justinmind som er interaktiv, noe har hjulpet oss med å illustrere løsningen mellom hverandre først. For det andre mellom oss og produkteieren.

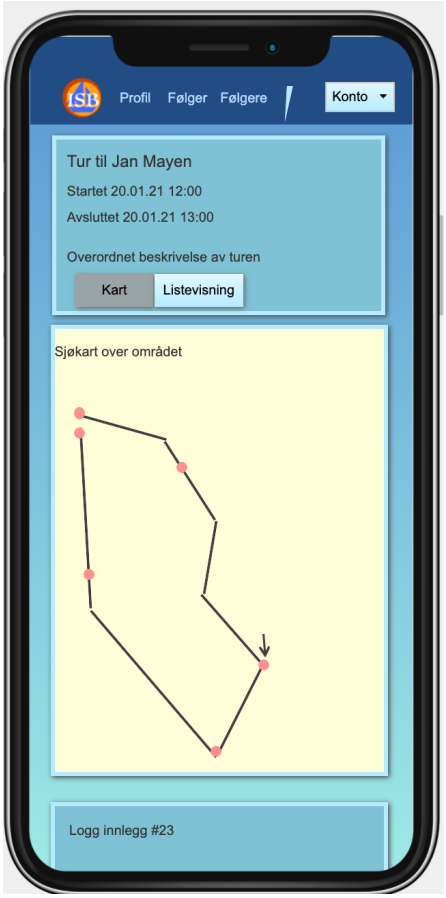

Profil
Følger
Følgere
Konto
Pågående turer
Tur til Jan Mayen
20.01.21
Ola Nordmann
epost@epost.com
75168416
Rediger profil
Registrer ny båt
Victoria
Victoria
Regnr 444444442
Båteiere
Start ny tur
Rediger båtinfo
Alle turer
Søke i tidligere turer


Rediger profil
Victoria
Victoria
Regnr 444444442
Båteiere
Start ny tur
Rediger båtinfo
Alle turer
Søke i tidligere turer
Tur til Jan Mayen
20.01.21
Tur til Shetland
20.01.21
Tur til Shetland
20.01.21
Tur til Shetland
20.01.21


Hei {bruker}
Velkommen til I samme båt
Vil du registrere en båt nå?
Registrer
Senere

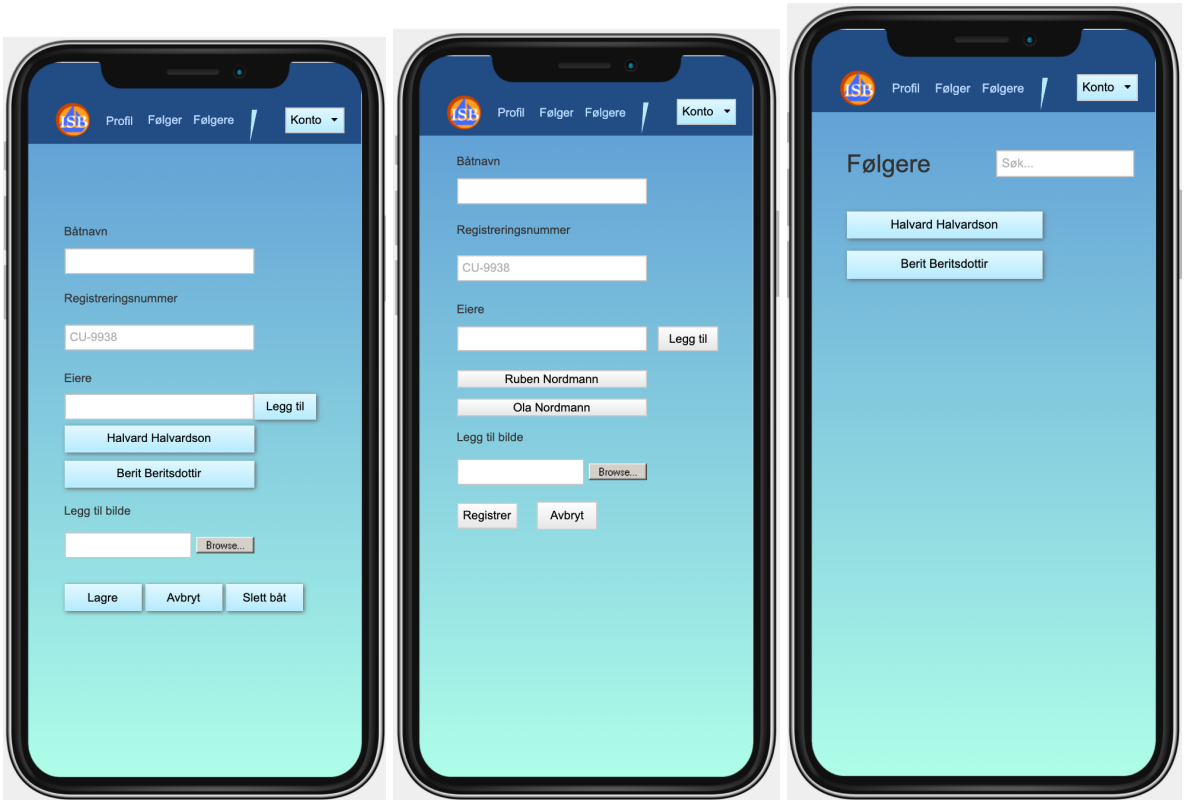

Profil
Følger
Følgere
Konto
Tur til Jan Mayen
Rediger
Dette er en liten dagstur til Jan Mayen
Avslutt tur
Jan Mayen
Hastighet
2 knop
Skriv logg
Distanse
20 nautiske mil

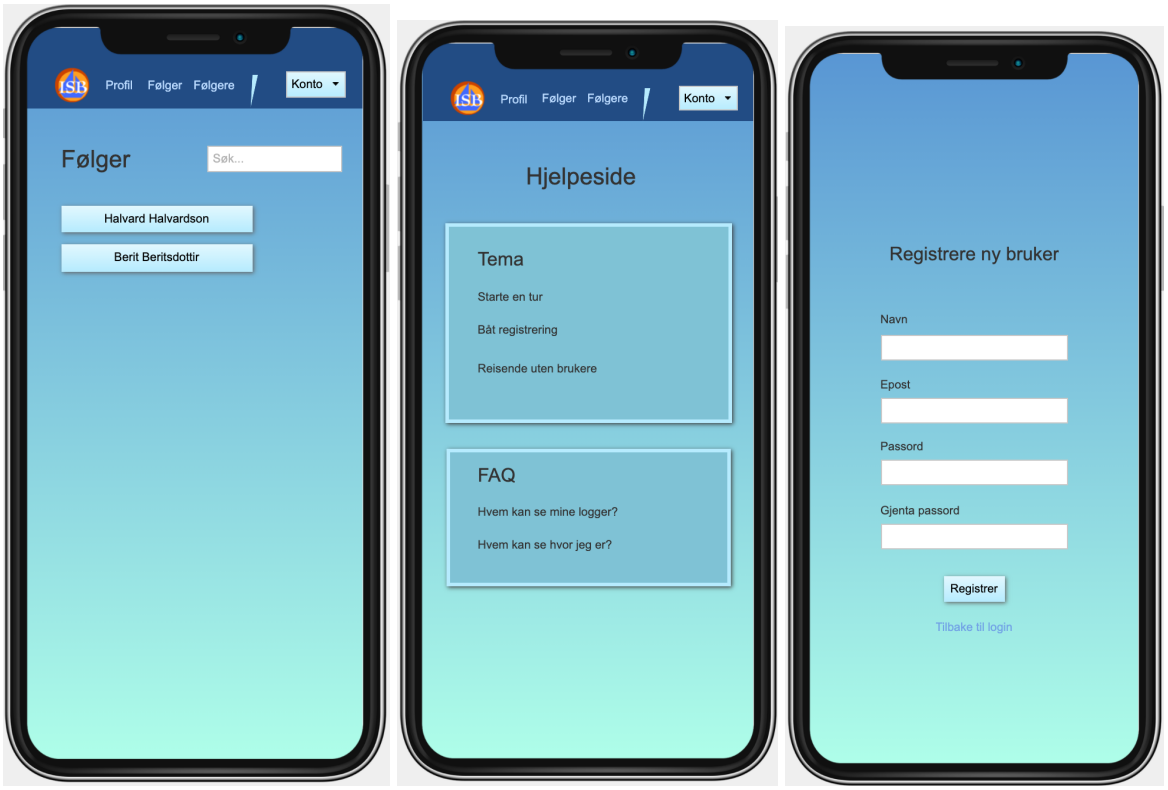

Profil
Følger
Følgere
Konto
Tittel
Innlegg
Legg til bilder
Browse...
Nåværende
Tidspunkt
Kart
Lagre

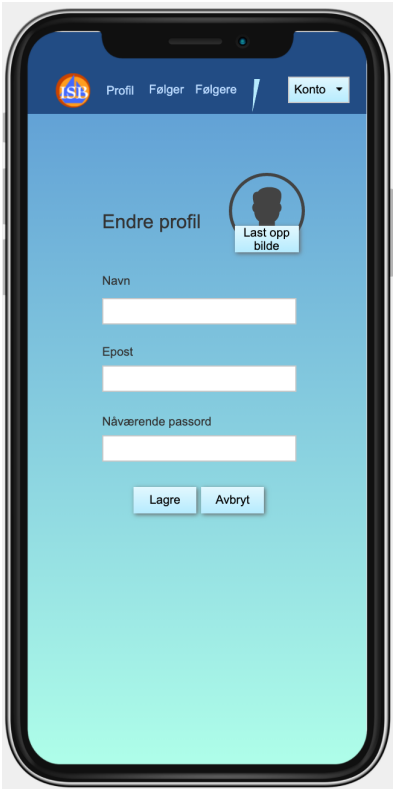

Profil
Følger
Følgere
Konto
Tur til Jan Mayen
Startet 20.01.21 12:00
Avsluttet 20.01.21 13:00
Overordnet beskrivelse av turen
Kart
Listevisning
Sjøkart over området
Logg innlegg #23

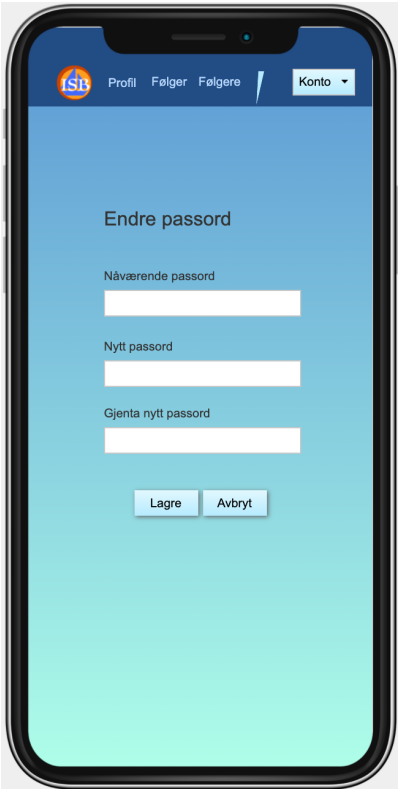

Profil
Følger
Følgere
Konto
Tur til Jan Mayen
Startet 20.01.21 12:00
Avsluttet 20.01.21 13:00
Overordnet beskrivelse av turen
Kart
Listevisning
Logg innlegg #22
Fre 20.02.21 13:14
Logg innlegg #23
Her så vi mange delfiner hoppe forbi
Fre 20.02.21 13:23

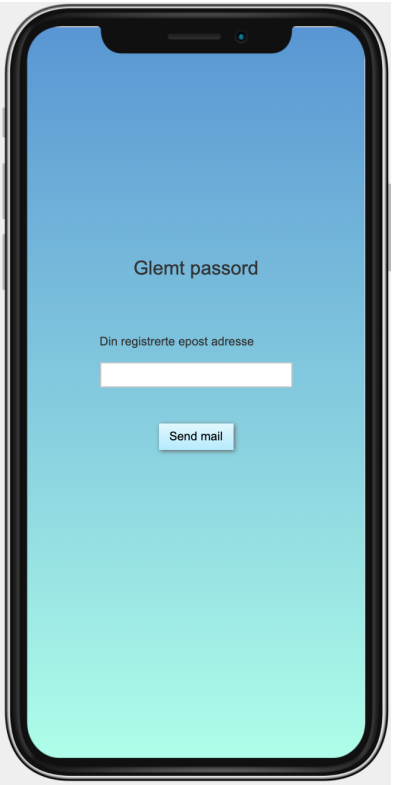

Profil
Følger
Følgere
Konto
Ola Nordmann
epost@epost.com
75168416
Victoria
Victoria
Regnr 444444442
Båteiere
Alle turer
Søke i tidligere turer
Tur til Jan Mayen
20.01.21
Tur til Shetland
20.01.21
Tur til Shetland
20.01.21
Tur til Shetland

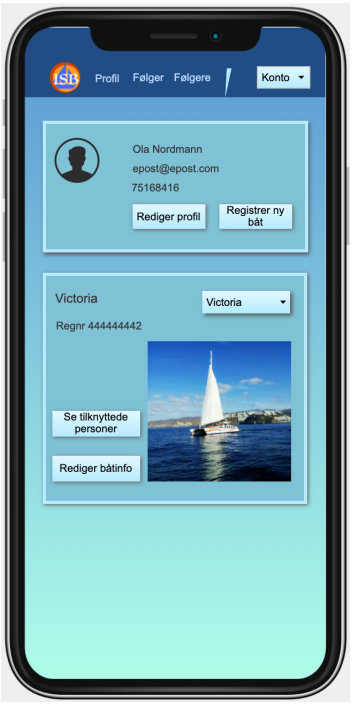

Profil
Følger
Følgere
Konto
Båtnavn
Registreringsnummer
CU-9938
Eiere
Legg til
Halvard Halvardson
Berit Beritsdottir
Legg til bilde
Browse...
Lagre
Avbryt
Slett båt
Ruben Nordmann
Ola Nordmann
Registrer
Følgere
Søk...


Profil
Følger
Følgere
Konto
Følger
Søk...
Halvard Halvardson
Berit Beritsdottir
Hjelpeside
Tema
Starte en tur
Båt registrering
Reisende uten brukere
FAQ
Hvem kan se mine logger?
Hvem kan se hvor jeg er?
Registrere ny bruker
Navn
Epost
Passord
Gjenta passord
Registrer
Tilbake til login


Profil
Følger
Følgere
Konto
Endre profil
Last opp bilde
Navn
Epost
Nåværende passord
Lagre
Avbryt


Profil
Følger
Følgere
Konto
Endre passord
Nåværende passord
Nytt passord
Gjenta nytt passord
Lagre
Avbryt


Glemt passord
Din registrerte epost adresse
Send mail


Profil
Følger
Følgere
Konto
Ola Nordmann
epost@epost.com
75168416
Rediger profil
Registrer ny båt
Victoria
Victoria
Regnr 444444442
Se tilknyttede personer
Rediger båtinfo

8.4 Visjonsdokument

8.4.1 Visjonsdokument (Første versjon)

Visjonsdokument – I samme båt

Anvendt Informasjonsteknologi, IDI, NTNU

Bacheloroppgave vår 2021

Revisjonshistorie

Dato	Versjon	Beskrivelse	Forfatter
13.01.2021	1.0	Generell skriving	Mahmoud Ibrahim, Eirik Plahte og Christian Riksvold
21.01.2021	1.1	Oppdaterte funksjoner og ikke-funksjonelle krav etter møte med oppdragsgiver	Mahmoud Ibrahim, Eirik Plahte og Christian Riksvold

Innholdsfortegnelse

1. Innledning

Dette dokumentet beskriver overordnede til bacheloroppgave “i samme båt” i emnet TDAT 3001 Bacheloroppgave Dataingeniør vår 2021. Oppgaven består i å utvikle en applikasjon “i samme båt”. Prosjektet skal utføres av 3.år dataingeniørstudenter i vårsemesteret 2021

2. Sammendrag problem og produkt

2.1 Problemsammendrag

Loggboka er en viktig komponent i et båthold. Den dekker turbeskrivelser, praktisk informasjon med mer, og kan fungere som "hyttebok". Det finnes i dag ingen gode digitale løsninger som dekker alle disse aspektene på en god måte, og ingen som ivaretar det sosiale aspektet som er til stede i båtlivet. Oppgaven er å utvikle en app som har som kjernefunksjonalitet å logge en båtreise med posisjonsspor, og ta tekst og bilder under veis som knyttes til spesielle hendelser. Innholdet skal kunne deles med medreisende og andre.

Problem med	dagens digitale loggbok-løsninger
berører	båteiere, mannskap og gjester på båt
som resultatet av dette	har man ingen løsninger som ivaretar det sosiale aspektet som er til stede i båtlivet, og ingen løsninger som dekker alle andre aspekter (f.eks. turbeskrivelser, praktisk informasjon med mer) på en tilfredsstillende måte
en vellykket løsning vil	dekke alle de tidligere nevnte aspektene

2.2 Produktsammendrag

“I samme båt” er en applikasjon som er ment for å tillate båteiere å logge en båtreise med nødvendig informasjon. Applikasjonen er brukerbasert og skal gi mulighet for å dele logger mellom brukere. Videre skal brukeren kunne legge til informasjon om hendelser i løpet av turen i form av tekst og/eller bilder. Båteieren skal kunne administrere hvem som har innsyn i båtens logg og hvem som kan logge.

For	Båteiere, mannskap og gjester på båt
som	Har behov for å holde en oversiktlig logg for båt og å kunne dele turer med andre utvalgte
I samme båt	En native applikasjon for Android

som	Tilbyr en intuitiv måte å logge båtens posisjon, hastighet, informasjon om turen, legge ved bilder og kunne dele det med andre utvalgte
I motsetning til	Andre løsninger som ikke prioriterer de sosiale aspektet og fokuserer heller på å rapportere om tekniske hendelser i båten.
Har vårt produkt	Et vennesystem som gjør det enkelt å dele hendelser og turer og gjør det enkelt for flere å logge på samme båt

3. Overordnet beskrivelse av interessenter og brukere

3.1 Oppsummering interessenter

Navn	Utdypende beskrivelse	Rolle under utviklingen
Donn Morrison	Førstelektor ved NTNU og veileder for prosjektet	Han skal delta på veiledningsmøter med studentene ca. hver 2. uke, og eventuelt ved andre anledninger dersom det er behov for det. Han skal også være med på å sensurere oppgaven.
Håvard Wigtil	Ansatt ved Kantega og oppdragsgiver	Han skal komme med krav til produktet og følge opp utviklingen underveis ved å ha møte med studentene hver uke. Han vil også ha en vesentlig rolle under testing av produktet og skal evt. hjelpe å finne andre testbrukere.
Mahmoud Ibrahim, Christian Riksvold, og Eirik Plahte	Studenter ved NTNU	Utviklere av produktet. Har hovedansvar for framgangen av prosjektet og dokumentasjonen.

3.2 Oppsummering brukere

Navn	Utdypende beskrivelse	Rolle under utviklingen	Representert av

Båteiere	Båteierne har hovedansvaret for å oppdatere og administrere loggen	Denne brukergruppen vil være med på å teste produktet i løpet av utviklingen	seg selv
Medreisende	Privatpersoner som er med på turen. For eksempel gjester av eierne eller turister	Ingen	*seg selv*
Mannskap	Mannskap på båt	Ingen	*seg selv*

4. Brukermiljøet

4.1 Sammendrag av brukernes behov

Kolonnet "Påvirker" peker på produktets funksjonelle egenskaper som blir påvirket av behovet.

Nr	Behov	Prioritet	Påvirker	Dagens løsning	Foreslått løsning
1	Kunne lagre informasjon mellom hver brukersesjon	Høy	Alle	Det finnes flere digitale notatblokker som kan brukes for å holde logg over båter	Brukeren skal kunne registrere en ny bruker og logge seg inn med en gyldig e-postadresse og passord. Slik vil appen kunne holde kontroll over informasjon brukeren legger til

2	Å se hvor man er på et passende kart	Høy	3, 4, 9, 12, 14	Andre båtapper har sjøkart som viser din og andres posisjon	Posisjonssporing integrert med sjøkart som oppdaterer posisjonen hvert minutt
3	Å kunne lagre reiser med posisjonssporing	Høy	4, 8, 9, 12, 14	Det finnes flere apper som kan kartlegge reiser	Løpende registrering av posisjon vha. mobiltelefon
4	Å logge spesifikke hendelser	Høy	5, 6, 7, 11, 12	Det finnes andre løsninger men de fleste fokuserer ikke på det sosiale aspektet og har heller ikke mulighet til å knytte bilder og informasjon til spesifikke posisjoner underveis på turen	Brukeren skal kunne legge til en hendelse med tekst og evt. bilde og knytte denne hendelsen til posisjonen brukeren er på

5	Kunne knytte personer/brukere til en reise	Høy	6, 12	Det finnes logg-apper hvor man kan skrive hvem som var med i tekst, men ingen som kan knytte sammen brukere. Alternativt kan sosiale medier som facebook brukes for dette behovet	Brukeren skal kunne legge til personer som er med på en reise enten de har egne brukere i appen eller ikke
6	La andre personer logge under samme tur	Høy	12	Ingen digitale løsninger finnes hvor flere personer kan logge under samme reise	Brukeren skal kunne administrere hvilke andre brukere som skal ha mulighet til å logge ved en spesifikk tur
7	Dele en logg med andre brukere	Høy	12	Her finnes alternative apper som kan dele båtlogger både i appen selv og på sosiale medier, disse blir nevnt senere i punkt 3.5	Applikasjonen skal ha et system som gjør at man kan dele reiser/logger med andre

8	Å se båtens hastighet	Høy	12	Google Maps har en speedometer-funksjon, men å ha dette integrert i appen gjør ting lettere for brukeren	Når appen ikke kjører i bakgrunnen, oppdaterer vi posisjonen ganske ofte (kanskje med noen sekunders mellomrom). Appen regner ut avstandene mellom hvert punkt, og kan dermed regne ut båtens hastighet
9	Å se hvor langt man har seilt	Høy	12	Flere andre båtapper har denne funksjonaliteten som vil fungere uansett om du har dekning eller ikke på deler av turen	Båtens posisjon oppdateres én gang hvert minutt og vil legge sammen avstandene mellom hvert punkt fra starten av reisen
10	Hjelp eller veiledning dersom det er noe i appen man ikke forstår	Høy	12	Flere liknende apper har integrert brukerhjelp	Appen har enten en brukerhåndbok eller pop-ups som dukker opp ved første gangs bruk, med mulighet for å se dem senere ved behov

11	Ha gode og enkle koblinger til personene/båtene man kjenner	Middels	5, 6, 7, 12, 14	Flere båtapper finnes hvor man kan ha venner/følgere som kan se loggen til hverandre	Brukeren kan følge venner og ha en venneliste som enten knyttes sammen via båter eller brukere
12	Vil se samme informasjonen som på appen med en annen maskin	Lav	Ingen	Mange konkurrerende apper har tilknyttede nettsider som gir et tilsvarende grensesnitt som appen	Applikasjonen vil ha en tilkoblet nettside som tilbyr en god representasjon av tilsvarende informasjon i appen
13	Å se hvor mye vann og olje/diesel man har brukt, og hvor mye man har igjen	Lav	12	Det finnes mange apper som holder kontroll over det tekniske aspektet ved båten	Applikasjonen kan samle inn data om drivstoff og vann ved hjelp av sensorer
14	Ønske å se om noen andre båter man har kjennskap til er i nærheten	Lav	12	Det finnes lignende systemer for ulike typer apper hvor brukere som er i nærheten av hverandre blir varslet	Dersom en venn av brukeren befinner seg innen en viss radius sendes et varsel til begge brukerne

15	Kunne uttrykke seg om noen andres logg	Lav	12	Det finnes mange måter å uttrykke seg på i reaksjon av andres logg, men det er ikke så mange andre båtapper som har dette integrert	Bruker skal kunne "like" og kommentere på logger brukeren har tilgang til å se

5. Alternativer til vårt produkt

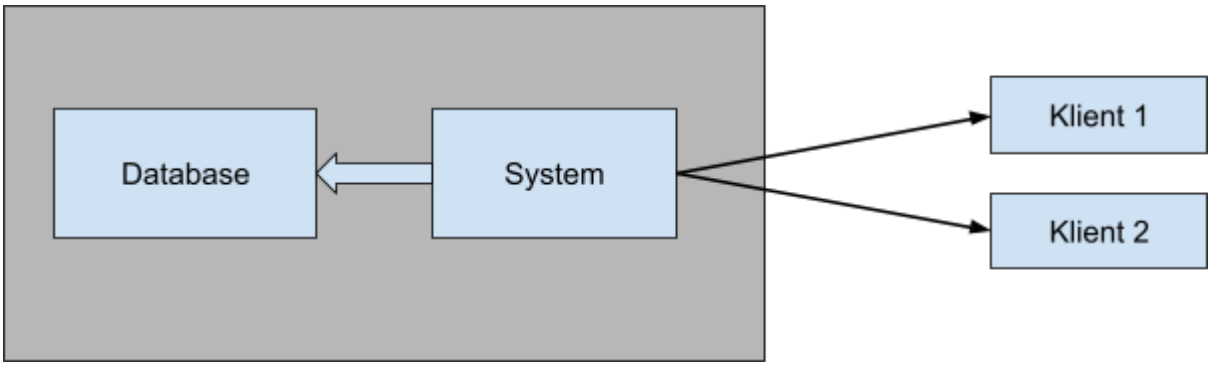
Det eksisterer noen alternativer til slike applikasjoner i dag, for eksempel "Blue Boat Log" og "Sailsense". Sistnevnte kan dele turer på sosiale medier, men fokuserer mest på å gi en oversikt over den tekniske tilstanden på båten uansett hvor du befinner deg. "Blue Boat Log" har lignende beskrivelse til det "I samme båt" tenker å ha, men har ikke like avansert loggesystem som knytter bilder og tekst til spesifikke posisjoner på turen. Disse appene har heller ikke funksjonalitet for at flere brukere kan skrive logg sammen.

Appen "Sail Expert" har et ganske avansert loggesystem, hvor man på detaljert vis kan planlegge reiser med hva slags utstyr som er med og hvem som er med og hva slags rolle de har. I tillegg har denne appen annen funksjonalitet, slik som integrert kompass. Appen har imidlertid ikke noe sosialt aspekt annet enn at man kan dele turer på andre sosiale medier, og man kan heller ikke legge til bilder underveis.

6. Produktoversikt

Under dette skal vi gå gjennom hensikten til omgivelsene og vi tar utgangspunktet i brukermiljø (4.1)

6.1 Forutsetninger og avhengigheter



Produktet krever at brukeren har en mobiltelefon som støtter Android, tilgang til internett og en gyldig e-postadresse for å opprette en ny bruker i systemet. Applikasjonen må ha tilgang til mobilens GPS for å logge reiser. I tillegg må man gi appen tilgang til kamera og/eller galleri dersom man vil legge til bilder.

6.2 Produktets funksjonelle egenskaper

Nr	Funksjon	Prioritet	Behov	Beskrivelse
1	Registrere bruker	Høy (10/10)	1	Klikker på registrerer en ny bruker med e-post og passord og ser at den ble registrert
2	Innlogging	Høy (10/10)	1	Logge inn ved bruk av registrert e-postadresse og passord
3	Posisjonssporing integrert med sjøkart	Høy (10/10)	2	Brukeren åpner kart på selve appen og kan se hvor båten ligger i sanntid
4	Lagre reiser med posisjonssporing	Høy (10/10)	3	Posisjonen oppdateres én gang i minuttet, og lagres i en database. Man kan da i ettertid se hele reisen på et kart
5	Registrere båt	Høy (10/10)	3	Man må registrere en båt for å kunne starte en tur

6	Fylle inn overordnet informasjon om turen	Høy (10/10)	4	I tillegg til å kunne logge fritt gjennom hele turen må brukeren fylle inn overordnet informasjon som gjelder for hele turen gjerne ut ifra en gitt mal
7	Knytte tekst og bilde til en posisjon	Høy (8/10)	4	Man skal kunne legge til hendelser i loggen, og disse hendelsene blir lagt til som tekst og evt. bilde. Disse hendelsene skal være knyttet til posisjonen man befinner seg når man logger dem
8	Legge ved brukere og navn som medreisende i en logg	Høy (8/10)	5	Båteier kan knytte andre brukere og medreisende til en logg. Dersom de som blir lagt til ikke er brukere, vil kun navnene deres være knyttet til loggen
9	Administrere hvem som kan logge ved en tur	Høy (8/10)	6	Båteier/admin kan bestemme hvem av de som er med på turen som kan logge underveis
10	Administrere hvem som kan se en logg	Høy (8/10)	7	Båteier/admin kan bestemme hvem som har innsyn i loggen, enten det er medreisende eller ikke
11	Dele en logg med en spesifikk bruker	Høy (7/10)	7	Båteier/admin skal ha muligheten til å dele loggen sin med spesifikke brukere. Implementeres enten gjennom et meldingssystem eller notifikasjoner
12	Brukerhåndbok eller integrert hjelp i appen	Høy (9/10)	10	Et hjelpeverktøy om man står fast eller ikke vet helt hvordan en utfører ulike handlinger i appen
13	Se andre båters logg	Høy (8/10)	7	Kunne se andres logg man får tilgang til
14	Måle og vise fart	Høy (7/10)	8	Når appen ikke kjører i bakgrunnen oppdaterer vi posisjonen ofte (med noen sekunders mellomrom), og ved å måle avstandene mellom hvert punkt blir det da mulig å regne ut farten til båten

15	Måle og vise seilt distanse	Høy (7/10)	9	Posisjonen til båten oppdateres én gang hvert minutt. Appen skal måle avstandene mellom hvert punkt, og finner dermed seilt distanse
16	Å skrive logg til en tidligere passert posisjon eller et tidligere tidspunkt	Høy (7/10)	4	Kunne legge til en hendelse på et punkt man var på kartet for en viss tid siden eller koble det til et tidspunkt tidligere. Uansett skal både tid og posisjon bli koblet til hendelsen.
17	Starte logging av tur etter at selve turen har startet, og likevel få med hele turen i loggen	Middels (6/10)	3	Hvis man glemmer å trykke på "Start ny tur" når man drar skal det være mulig å velge en startposisjon og et starttidspunkt for turen, slik at ruten man tar blir så komplett som mulig.
18	Manuelt bestemme endeposisjon og endetidspunkt for en tur	Middels (6/10)	3	Hvis man glemmer å trykke på "Avslutt tur" når man er ferdig skal det være mulig å velge en endeposisjon og tilsvarende endetidspunkt slik at ruten blir riktig.
19	Logge flere turer som en del av samme reise	Middels (6/10)	3	Hvis man skal på en lengre tur med flere etapper skal det være mulig å kategorisere flere turer som en del av samme reise, slik at man har full oversikt.
20	Kunne manøvrere ruter ut fra havner på kartet	Middels (6/10)	3	Kunne velge spesifikke havner som destinasjon eller avgangspunkt samt lagre sine egne private havner.
21	Tilbakestille passord om glemt	Middels (6/10)	1	Dersom man har glemt passordet sitt skal man få tilsendt en mail hvor man kan endre passordet
22	Redigere logg	Middels (6/10)	3	Det skal være mulig å redigere loggen dersom man skriver noe feil. Dette skal enten gjelde kun for enkelte hendelser som blir lagt til i loggen, eller for overordnet informasjon om turen i tillegg.

23	Søke i logger du har tilgang til basert på posisjon	Middels (5/10)	3	Kunne se turer som har blitt kjørt i nærheten (eller et spesifisert område) og søke gjennom dem. Dette gjelder dine egne logger og andre logger du har tilgang til.
24	Se egen og andres profiler	Middels (5/10)	1	Man skal kunne gå inn på egen og andres profiler, hvor det står diverse brukerinformasjon
25	Redigere brukerinformasjon	Middels (5/10)	1	Endre personlig informasjon samt passord til bruker
26	Redigere båt	Middels (5/10)	1	Man må kunne redigere båtene sine, f.eks. hvis det er noen nye deleiere eller legge til nytt bilde av båten
27	Følge en bruker / legge til venner i en venneliste	Middels (5/10)	11	Det skal være mulig å enten sende og godta venneforespørsler mellom brukere, eller å følge andre brukere
28	Følge en båt / legge til båt i en "venneliste" med båter	Middels (5/10)	11	Noen mulige funksjoner er at man kan følge en båt, slik alle innlegg som omhandler den båten dukker opp i en "feed", at man er "venner" med en båt, eller at båter er "venner" med hverandre
29	Kunne opprette en bruker og tilknytte gammel informasjon om ditt navn	Middels (5/10)	1	Om man har blitt tilknyttet reiser før man hadde bruker, skal man kunne velge om denne informasjonen skal knyttes til seg når det opprettes en ordentlig brukerprofil
30	Søke gjennom tidligere turer	Middels (4/10)	3	For å gjøre grensesnittet enklere burde man kunne søke gjennom tidligere gjennomførte turer
31	Bytte mellom ulike typer kart	Middels (4/10)	2	Kunne endre visning av kartet for å se detaljer på land eller virkelighetsbilder. (Sjøkart, flyfoto, landkart)

32	Søke gjennom venner	Middels (4/10)	11	For å gjøre grensesnittet enklere burde man kunne søke gjennom vennelisten sin
33	“Like” en logg	Middels (4/10)	15	En mulig utvidelsesfunksjon er at man kan “like” en logg, enten ved at loggen blir tilsendt en bruker, eller ved at brukeren finner den i en slags “feed” hvor vennene til brukeren legger ut loggen sin
34	Kommentere en logg	Lav (3/10)	15	En mulig utvidelsesfunksjon er at man kan kommentere en logg, enten ved at loggen blir tilsendt en bruker, eller ved at brukeren finner den i en slags “feed” hvor vennene til brukeren legger ut loggen sin
35	Logge inn med bruker fra andre sosiale medier som facebook eller google	Lav (2/10)	1	Brukeren skal ha mulighet til å bruke en allerede opprettet bruker fra et annet sosialt medium i stedet for å opprette en helt ny bruker
36	Sende meldinger til andre brukere	Lav (2/10)	11	En mulig utvidelse for å kunne kommunisere enklere med andre brukere uten behov for å benytte andre kommunikasjonsverktøy
37	Opprette grupper	Lav (1/10)	11	En mulig utvidelse er å opprette grupper med venner, bl.a. for å enkelt gi personer innsyn i en logg
38	Administrere hvem som kan se din posisjon	Lav (3/10)	14	Man må kunne bestemme hvem som skal se sin informasjon av personvernsmessige hensikter
39	Bli varslet om brukere i nærheten	Lav (3/10)	14	Om en bruker man har godtatt skal kunne se din posisjon er i nærheten skal begge brukerne få et varsel
40	Bedre visning av turer på tilknyttet nettside	Lav (2/10)	12	En mulig utvidelse er å ha en tilkoblet nettside som tilbyr en god representasjon av tilsvarende informasjon som det man har i appen

41	Spore forbruk (vann, olje, diesel)	Lav (2/10)	13	Enkel sporing av de viktigste kildene av drivstoff og andre nødvendigheter man trenger for en båttur. Spores enten ved sensorer eller kun ved manuell oppdatering

6.3 Ikke-funksjonelle egenskaper og andre krav

- *Alle data som benyttes av applikasjonen "I samme båt" skal lagres i database. Studentene og oppdragsgiver har avtalt at NTNUs mySQL-database skal brukes.*
- *Løsningen skal ha en god brukskvalitet, lett og intuitiv å bruke.*
- *Både server og klient skal ha gode loggsystemer hvor man kan få oversikt over hvilke koblinger som har blitt oppført og hvilke funksjoner som har blitt utført samt gode beskrivelser av eventuelle feil som kunne oppstå.*
- *Koden skal ha svært stor testdekning med både enhetstesting og integrasjonstesting med spesielt vekt på sistnevnte. Ingen spesifikk prosentandel er nevnt, men ca 90-95% ligger underforstått.*
- *Applikasjonen skal ikke lagre for mye informasjon av brukeren. Kun nødvendige detaljer, som i første utkast bare vil bestå av epost og navn. Om bruker vil dele sin posisjon med andre, må brukeren få klar beskjed om hva dette innebærer.*

Dette dokumentet er den endelige visjonsdokumentet-versjonen som er et resultat av scrum-metodikk vi brukte i prosjektet. Den versjonen har samme struktur som "visjonsdokument første versjon" (se punkt 8.4.1), men det er funksjoner som har blitt endret, lagt til og fjernet.

Visjonsdokument – I samme båt

Anvendt Informasjonsteknologi, IDI, NTNU

Bacheloroppgave vår 2021

Revisjonshistorie

Dato	Versjon	Beskrivelse	Forfatter
13.01.2021	1.0	Generell skriving	Mahmoud Ibrahim, Eirik Plahte og Christian Riksvold
21.01.2021	1.1	Oppdaterte funksjoner og ikke-funksjonelle krav etter møte med oppdragsgiver	Mahmoud Ibrahim, Eirik Plahte og Christian Riksvold
17.05.2021	2.0	Oppdaterte funksjoner og ikke-funksjonelle krav på slutten av prosjektet	Mahmoud Ibrahim, Eirik Plahte og Christian Riksvold

Innholdsfortegnelse

1. Innledning

Dette dokumentet beskriver overordnede til bacheloroppgave “i samme båt” i emnet TDAT 3001 Bacheloroppgave Dataingeniør vår 2021. Oppgaven består i å utvikle en applikasjon “i samme båt”. Prosjektet skal utføres av 3.år dataingeniørstudenter i vårsemesteret 2021. Dokumentet er er den endelige versjonen som viser alle endringer som har skjedd i løpet av utviklingsprosessen som er et resultatet av smidig utviklingsmetodikk vi har brukt.

2. Sammendrag problem og produkt

2.1 Problemsammendrag

Loggboka er en viktig komponent i et båthold. Den dekker turbeskrivelser, praktisk informasjon med mer, og kan fungere som "hyttebok". Det finnes i dag ingen gode digitale løsninger som dekker alle disse aspektene på en god måte, og ingen som ivaretar det sosiale aspektet som er til stede i båtlivet. Oppgaven er å utvikle en app som har som kjernefunksjonalitet å logge en båtreise med posisjonsspor, og ta tekst og bilder under veis som knyttes til spesielle hendelser. Innholdet skal kunne deles med medreisende og andre.

Problem med	dagens digitale loggbok-løsninger
berører	båteiere, mannskap og gjester på båt
som resultatet av dette	har man ingen løsninger som ivaretar det sosiale aspektet som er til stede i båtlivet, og ingen løsninger som dekker alle andre aspekter (f.eks. turbeskrivelser, praktisk informasjon med mer) på en tilfredsstillende måte
en vellykket løsning vil	dekke alle de tidligere nevnte aspektene

2.2 Produktsammendrag

“I samme båt” er en applikasjon som er ment for å tillate båteiere å logge en båtreise med nødvendig informasjon. Applikasjonen er brukerbasert og skal gi mulighet for å dele logger mellom brukere. Videre skal brukeren kunne legge til informasjon om hendelser i løpet av

turen i form av tekst og/eller bilder. Båteieren skal kunne administrere hvem som har innsyn i båtens logg og hvem som kan logge.

For	Båteiere, mannskap og gjester på båt
som	Har behov for å holde en oversiktlig logg for båt og å kunne dele turer med andre utvalgte
I samme båt	En native applikasjon for Android
som	Tilbyr en intuitiv måte å logge båtens posisjon, hastighet, informasjon om turen, legge ved bilder og kunne dele det med andre utvalgte
I motsetning til	Andre løsninger som ikke prioriterer de sosiale aspektet og fokuserer heller på å rapportere om tekniske hendelser i båten.
Har vårt produkt	Et vannsystem som gjør det enkelt å dele hendelser og turer og gjør det enkelt for flere å logge på samme båt

3. Overordnet beskrivelse av interessenter og brukere

3.1 Oppsummering interessenter

Navn	**Utdypende beskrivelse**	**Rolle under utviklingen**
Donn Morrison	Førstelektor ved NTNU og veileder for prosjektet	Han skal delta på veiledningsmøter med studentene ca. hver 2. uke, og eventuelt ved andre anledninger dersom det er behov for det. Han skal også være med på å sensurere oppgaven.
Håvard Wigtil	Ansatt ved Kantega og oppdragsgiver	Han skal komme med krav til produktet og følge opp utviklingen underveis ved å ha møte med studentene hver uke. Han vil også ha en vesentlig

		rolle under testing av produktet og skal evt. hjelpe å finne andre testbrukere.
Mahmoud Ibrahim, Christian Riksvold, og Eirik Plahte	Studenter ved NTNU	Utviklere av produktet. Har hovedansvar for framgangen av prosjektet og dokumentasjonen.

3.2 Oppsummering brukere

Navn	Utdypende beskrivelse	Rolle under utviklingen	Representert av
Båteiere	Båteierne har full kontroll til å administrere båten, turene og logginnleggene	Denne brukergruppen vil være med på å teste produktet i løpet av utviklingen	seg selv
Passasjerer	Personer som er med om bord på turen.	Ingen	*seg selv*
Bekjente	Personer som gjerne blir delt turene med, men ikke nødvendigvis er med på tur.	Ingen	*seg selv*

4. Brukermiljøet

4.1 Sammendrag av brukernes behov

Kolonnet “Påvirker” peker på produktets funksjonelle egenskaper som blir påvirket av behovet.

Nr	Behov	Prioritet	Påvirker	Dagens løsning	Foreslått løsning
1	Kunne lagre informasjon mellom hver brukersesjon	Høy	Alle	Det finnes flere digitale notatblokker som kan brukes for å holde logg over båter	Brukeren skal kunne registrere en ny bruker og logge seg inn med en gyldig e-postadresse og passord. Slik vil appen kunne holde kontroll over informasjon brukeren legger til
2	Å se hvor man er på et passende kart	Høy	3, 4, 9, 12, 14	Andre båt-apper har sjøkart som viser din og andres posisjon	Posisjonssporing integrert med sjøkart som oppdaterer posisjonen hvert 15 meter
3	Å kunne lagre reiser med posisjonssporing	Høy	4, 8, 9, 12, 14	Det finnes flere apper som kan kartlegge reiser	Løpende registrering av posisjon vha. mobiltelefon
4	Å logge spesifikke hendelser	Høy	8, 9, 18, 19,	Det finnes andre løsninger men de fleste fokuserer ikke på det sosiale aspektet og har heller ikke mulighet til å knytte bilder og informasjon til spesifikke posisjoner underveis på turen	Brukeren skal kunne legge til en hendelse med tekst og evt. bilde og knytte denne hendelsen til posisjonen brukeren er på
5	Kunne knytte personer/brukere til en reise	Høy	12	Det finnes logg-apper hvor man kan skrive hvem som var med	Brukeren skal kunne legge til personer som er med på

				i tekst, men ingen som kan knytte sammen brukere. Alternativt kan sosiale medier som facebook brukes for dette behovet	en reise enten de har egne brukere i appen eller ikke
6	La andre personer logge under samme tur	Høy	12	Ingen digitale løsninger finnes hvor flere personer kan logge under samme reise	Brukeren skal kunne administrere hvilke andre brukere som skal ha mulighet til å logge ved en spesifikk tur
7	Dele en logg med andre brukere	Høy	12, 13	Her finnes alternative apper som kan dele båtlogger både i appen selv og på sosiale medier, disse blir nevnt senere i punkt 3.5	Applikasjonen skal ha et system som gjør at man kan dele reiser/logger med andre
8	Å se båtens hastighet	Høy	16	Google Maps har en speedometer-funksjon, men å ha dette integrert i appen gjør ting lettere for brukeren	Når appen ikke kjører i bakgrunnen, oppdaterer vi posisjonen ganske ofte (kanskje med noen sekunders mellomrom). Appen regner ut avstandene mellom hvert punkt, og kan dermed regne ut båtens hastighet
9	Å se hvor langt man har seilt	Høy	17	Flere andre båtapper har denne funksjonaliteten	Båtens posisjon oppdateres én gang hvert minutt og vil legge sammen avstandene

				som vil fungere uansett om du har dekning eller ikke på deler av turen	mellom hvert punkt fra starten av reisen
10	Hjelp eller veiledning dersom det er noe i appen man ikke forstår	Høy	20	Flere liknende apper har integrert brukerhjelp	Appen har enten en brukerhåndbok eller pop-ups som dukker opp ved første gangs bruk, med mulighet for å se dem senere ved behov
11	Ha gode og enkle koblinger til personene/båtene man kjenner	Middels	5, 6, 7, 8, 9, 10, 11, 12,	Flere båtapper finnes hvor man kan ha venner/følgere som kan se loggen til hverandre	Brukeren kan følge venner og ha en venneliste som enten knyttes sammen via båter eller brukere
12	Vil se samme informasjonen som på appen med en annen maskin	Lav	55	Mange konkurrerende apper har tilknyttede nettsider som gir et tilsvarende grensesnitt som appen	Applikasjonen vil ha en tilkoblet nettside som tilbyr en god representasjon av tilsvarende informasjon i appen
13	Å se hvor mye vann og olje/diesel man har brukt, og hvor mye man har igjen	Lav	56	Det finnes mange apper som holder kontroll over det tekniske aspektet ved båten	Applikasjonen kan samle inn data om drivstoff og vann ved hjelp av sensorer
14	Ønske å se om noen andre	Lav	53, 54	Det finnes lignende systemer	Dersom en venn av brukeren befinner seg innen

	båter man har kjennskap til er i nærheten			for ulike typer apper hvor brukere som er i nærheten av hverandre blir varslet	en viss radius sendes et varsel til begge brukerne
15	Kunne uttrykke seg om noen andres logg	Lav	49, 51	Det finnes mange måter å uttrykke seg på i reaksjon av andres logg, men det er ikke så mange andre båtapper som har dette integrert	Bruker skal kunne "like" og kommentere på logger brukeren har tilgang til å se

5. Alternativer til vårt produkt

Det eksisterer noen alternativer til slike applikasjoner i dag, for eksempel "Blue Boat Log" og "Sailsense". Sistnevnte kan dele turer på sosiale medier, men fokuserer mest på å gi en oversikt over den tekniske tilstanden på båten uansett hvor du befinner deg. "Blue Boat Log" har lignende beskrivelse til det "I samme båt" tenker å ha, men har ikke like avansert loggesystem som knytter bilder og tekst til spesifikke posisjoner på turen. Disse appene har heller ikke funksjonalitet for at flere brukere kan skrive logg sammen.

Appen "Sail Expert" har et ganske avansert loggesystem, hvor man på detaljert vis kan planlegge reiser med hva slags utstyr som er med og hvem som er med og hva slags rolle de har. I tillegg har denne appen annen funksjonalitet, slik som integrert kompass. Appen har imidlertid ikke noe sosialt aspekt annet enn at man kan dele turer på andre sosiale medier, og man kan heller ikke legge til bilder underveis.

6. Produktoversikt

Under dette skal vi gå gjennom hensikten til omgivelsene og vi tar utgangspunktet i brukermiljø (4.1)

6.1 Forutsetninger og avhengigheter

Produktet krever at brukeren har en mobiltelefon som støtter Android, tilgang til internett og en gyldig e-postadresse for å opprette en ny bruker i systemet. Applikasjonen må ha tilgang til mobilens GPS for å logge reiser. I tillegg må man gi appen tilgang til kamera og/eller galleri dersom man vil legge til bilder. nr funksjon prio behov beskrivelse

6.2 Produktets funksjonelle egenskaper

Nr	Funksjon	Prioritet	Behov	Beskrivelse
1	Registrere bruker	Høy (10/10)	1	Klikker på registrerer en ny bruker med e-post og passord og ser at den ble registrert
2	Logge inn	Høy (10/10)	1	Logge inn ved bruk av registrert e-postadresse og passord
3	Spore en tur med gps	Høy (10/10)	2	Brukeren skal kunne starte en tur og si at appen skal spore ruta hens
4	Se en tur på sjøkart	Høy (10/10)	2	Brukeren skal kunne se ruta si på et sjøkart
5	Lagre reiser med posisjonssporing	Høy (10/10)	3, 11	Posisjonen oppdateres én gang i minuttet, og lagres i en database. Man kan da i ettertid se hele reisen på et kart
6	Registrere båt	Høy (10/10)	11	Man må registrere en båt for å kunne starte en tur
7	Fylle inn overordnet informasjon om turen	Høy (10/10)	3, 11	I tillegg til å kunne logge fritt gjennom hele turen må brukeren fylle inn overordnet informasjon som gjelder for hele turen gjerne ut ifra en gitt mal

8	Knytte tekst og bilde til en posisjon	Høy (8/10)	4, 11	Man skal kunne legge til hendelser i loggen, og disse hendelsene blir lagt til som tekst og evt. bilde. Disse hendelsene skal være knyttet til posisjonen man befinner seg når man logger dem
9	Se alle logginnlegg som hører til en tur	Høy (8/10)	4, 11	Inne på siden til en tur skal man kunne se innleggene i listeform og som markører på kartet
10	Se tidligere turer ut fra en liste	Høy (8/10)	3, 11	Man skal ha en liste over tidligere turer, og man skal kunne trykke på en spesifikk tur og få informasjon om denne turen
11	Se oversikt over båter	Høy (8/10)	1, 11	Automatisk komme inn på en profil med visning av aktiv båt og de nyligste turene
12	Legge ved brukere og navn som medreisende i en tur	Høy (8/10)	5, 11	Båteier kan knytte andre brukere og medreisende til en tur. Dersom de som blir lagt til ikke er brukere, vil kun navnene deres være knyttet til turen
13	Dele en tur med en spesifikk bruker	Høy (7/10)	6	Båteier/admin skal ha muligheten til å dele turen sin med spesifikke brukere. Implementeres enten gjennom et meldingssystem eller notifikasjoner
14	Kunne få hjelp om man ikke forstår appens oppsett	Høy (9/10)	10	Et hjelpeverktøy om man står fast eller ikke vet helt hvordan en utfører ulike handlinger i appen
15	Se andre båters turer	Høy (7/10)		Kunne se andres turer med tilhørende logginnlegg man får tilgang til

16	Se farten til båten i sanntid	Høy (7/10)	8	Når appen ikke kjører i bakgrunnen oppdaterer vi posisjonen ofte (med noen sekunders mellomrom), og ved å måle avstandene mellom hvert punkt blir det da mulig å regne ut farten til båten
17	Se distansen til turen underveis	Høy (7/10)	9	Posisjonen til båten oppdateres én gang hvert minutt eller oftere. Appen skal måle avstandene mellom hvert punkt, og finner dermed seilt distanse
18	Skrive logginnlegg til en tidligere passert posisjon	Høy (7/10)	4	Kunne legge til en hendelse på et punkt man var på kartet for en viss tid siden. Både tid og posisjon skal bli koblet til hendelsen.
19	Skrive logginnlegg til et tidligere tidspunkt	Høy (7/10)	4	Kunne legge til en hendelse på et tidligere tidspunkt. Både tid og posisjon skal bli koblet til hendelsen.
20	Få hjelp til å komme i gang	Middels (6/10)	10	Første gang du logger inn skal du få en egen velkomstskjerm som hjelper deg å registrere båt om du vil og kanskje gi et lite innblikk i hvordan appen fungerer generelt.
21	Redigere informasjon om tur	Middels (6/10)	3	Etter å ha fylt inn informasjon om turen burde det være mulig å redigere den om noe ble fylt inn feil.
22	Starte sporing av tur etter at selve turen har startet, og likevel få med hele turen i loggen	Middels (6/10)	3	Hvis man glemmer å trykke på “Start ny tur” når man drar skal det være mulig å velge en startposisjon og et starttidspunkt for turen, slik at ruten man tar blir så komplett som mulig.

23	Manuelt bestemme endeposisjon og endetidspunkt for en tur	Middels (6/10)	3	Hvis man glemmer å trykke på “Avslutt tur” når man er ferdig skal det være mulig å velge en endeposisjon og tilsvarende endetidspunkt slik at ruten blir riktig.
24	Arkivere flere turer som en del av samme reise	Middels (6/10)	3	Hvis man skal på en lengre tur med flere etapper skal det være mulig å kategorisere flere turer som en del av samme reise, slik at man har full oversikt.
25	Legge til havn på kartet manuelt	Middels (6/10)		Kunne legge til en havn ved å trykke på et kart og velge posisjonen og navn selv.
26	Legge til havn automatisk på avgangspunkt og ankomstpunkt etter en tur	Middels (6/10)		Om en havn ikke befinner seg i en satt radius hvor en tur starter eller slutter, skal det bli opprettet en havn der automatisk.
27	Redigere både navn og posisjon til en eksisterende havn	Middels (6/10)		Endre navn og posisjon på en havn om den blir plassert litt feil i utgangspunktet.
28	Slette en havn	Middels (6/10)		Om en havn ikke er i bruk lenger eller har blitt opprettet ved en feil skal den kunne slettes.
29	Se havner på kart	Middels (6/10)		Man burde kunne se havner på kart, både mens man er på tur, og ellers, slik at man har oversikt over steder man har vært innom på tvers av alle turer

30	Tilbakestille passord om glemt	Middels (6/10)	1	Dersom man har glemt passordet sitt skal man få tilsendt en mail hvor man kan endre passordet
31	Redigere logginnlegg	Middels (6/10)	4	Brukeren som skrev logginnlegget skal kunne redigere det etter publisering
32	Slette logginnlegg	Middels (6/10)	4	Brukeren som skrev logginnlegget skal kunne slette det etter publisering
33	Få varsel	Middel (5/10)	5	Man skal få varsel dersom en bruker deler en tur med deg eller har lagt deg til som tilknyttet en båt, osv.
34	Bekrefte å bli lagt til på en tur eller båt	Middels (5/10)	5	Når man får varsel om at at man er lagt til i en tur/båt skal manbekrefte at man er med, deretter får man mulighet til å logge
35	Redigere logginnlegg man ikke har skrevet om man er admin	Middels (5/10)	4	Båteier skal kunne redigere alle innleggene på turen hen startet
36	Slette logginnlegg man ikke har skrevet om man er admin	Middels (5/10)	4	Båteier skal kunne slette alle innleggene på turen hen startet
37	Slette tur	Middels (5/10)	3	Båteier skal kunne slette turer
38	Slette båt	Middels (5/10)		Brukeren skal kunne slette båten sin

39	Søke etter logginnlegg i nærheten	Middels (5/10)		Kunne se turer som har blitt kjørt i nærheten (eller et spesifisert område) og søke gjennom dem. Dette gjelder dine egne logger og andre logger du har tilgang til.
40	Se andres profiler	Middels (5/10)		Man skal kunne gå inn på andres profiler, hvor det står diverse brukerinformasjon
41	Redigere brukerinformasjon	Middels (5/10)	1	Endre personlig informasjon til brukeren som navn, mail og profilbilde
42	Endre passord	Middels (5/10)	1	Endre passord er en egen funksjon med ekstra sikkerhet
43	Redigere båt	Middels (5/10)		Man må kunne redigere båtene sine, f.eks. hvis det er noen nye deleiere eller legge til nytt bilde av båten
44	Legge til tilknyttede til båt	Middels (5/10)		Hvis man er fast tilknyttet en båt skal man kunne logge uavhengig om man er med på tur eller ikke
45	Kunne tilknytte gammel informasjon om ditt navn til din bruker	Middels (5/10)	1	Om man har blitt tilknyttet reiser før man hadde bruker, skal man kunne velge om denne informasjonen skal knyttes til seg når det opprettes en ordentlig brukerprofil
46	Legge til logginnlegg som tilhører en båt	Middels (5/10)	4	I forhold til logging, kunne logge når det ikke er en tur, for eksempel ved vedlikehold. Man må kunne logge uten at det blir koblet til en trip
47	Søke gjennom tidligere turer	Middels (4/10)	3	For å gjøre grensesnittet enklere burde man kunne søke gjennom tidligere gjennomførte turer
48	Bytte mellom ulike typer kart	Middels (4/10)	2	Kunne endre visning av kartet for å se detaljer på land eller virkelighetsbilder.

				(Sjøkart, flyfoto, landkart)
49	"Like" et logginnlegg	Middels (4/10)	15	En mulig utvidelsesfunksjon er at man kan "like" en logg, enten ved at loggen blir tilsendt en bruker, eller ved at brukeren finner den i en slags "feed" hvor vennene til brukeren legger ut loggen sin
50	Slette konto	Lav (3/10)	1	Brukeren kan slette kontoen og alle data som er lagret på den kontoen
51	Kommentere et logginnlegg	Lav (3/10)	15	En mulig utvidelsesfunksjon er at man kan kommentere en logg, enten ved at loggen blir tilsendt en bruker, eller ved at brukeren finner den i en slags "feed" hvor vennene til brukeren legger ut loggen sin
52	Logge inn med bruker fra andre sosiale medier som facebook eller google	Lav (2/10)	1	Brukeren skal ha mulighet til å bruke en allerede opprettet bruker fra et annet sosialt medium i stedet for å opprette en helt ny bruker
53	Administrere hvem som kan se din posisjon	Lav (3/10)	14	Man må kunne bestemme hvem som skal se sin informasjon av personvernmessige hensikter
54	Bli varslet om brukere i nærheten	Lav (3/10)	14	Om en bruker man har godtatt skal kunne se din posisjon er i nærheten skal begge brukerne få et varsel
55	Se alt fra applikasjonen på en tilknyttet nettside	Lav (2/10)	12	En mulig utvidelse er å ha en tilkoblet nettside som tilbyr en god representasjon av tilsvarende informasjon som det man har i appen

56	Spore forbruk (vann, olje, diesel)	Lav (2/10)	13	Enkel sporing av de viktigste kildene av drivstoff og andre nødvendigheter man trenger for en båttur. Spores enten ved sensorer eller kun ved manuell oppdatering

6.3 Ikke-funksjonelle egenskaper og andre krav

- Alle data som benyttes av applikasjonen "I samme båt" skal lagres i database. Studentene og oppdragsgiver har avtalt at NTNUs mySQL-database skal brukes.
- Løsningen skal ha en god brukskvalitet, lett og intuitiv å bruke.
- Både server og klient skal ha gode loggsystemer hvor man kan få oversikt over hvilke koblinger som har blitt oppført og hvilke funksjoner som har blitt utført samt gode beskrivelser av eventuelle feil som kunne oppstå.
- Koden skal ha svært stor testdekning med både enhetstesting og integrasjonstesting med spesielt vekt på sistnevnte. Ingen spesifikk prosentandel er nevnt, men ca 90-95% ligger underforstått.
- Applikasjonen skal ikke lagre for mye informasjon av brukeren. Kun nødvendige detaljer, som i første utkast bare vil bestå av epost og navn. Om bruker vil dele sin posisjon med andre, må brukeren få klar beskjed om hva dette innebærer.