Malene Lundemo
Anders Lunde Hagen

Svartelistede arter i vegkanten

Bacheloroppgave i programmering
Veileder: Peter Nussbaum
Mai 2023

Bacheloroppgave

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

NTNU
Kunnskap for en bedre verden

Malene Lundemo
Anders Lunde Hagen

Svartelistede arter i vegkanten

Bacheloroppgave i programmering
Veileder: Peter Nussbaum
Mai 2023

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

NTNU

Kunnskap for en bedre verden

Sammendrag av Bacheloroppgave

Tittel:	Svartelistede Arter i Vegkanten
Dato:	22.05.2023
Deltakere:	Malene Lundemo Anders Lunde Hagen
Veileder:	Peter Nussbaum
Oppdragsgiver:	Gjøvik kommune
Kontaktperson:	Ingun Revhaug
Nøkkelord:	Programmering, Kommune, Web, Relasjonsdatabase, Scrum
Antall sider:	64
Antall vedlegg:	4
Tilgjengelighet:	Åpen

Sammendrag: Gjøvik kommune er selv ansvarlig for å opprettholde naturmangfoldet, noe som inkluderer bekjemping av svartelistede plantearter. Kommunen etterspurte en løsning fra oss som ville forenkle registreringen og dermed bekjempelsen av svartelistede plantearter. Vår løsning inkluderte to nettsideløsninger som kommuniserer med hverandre. Den ene innhenter henvendelser fra innbyggere og besøkede i Gjøvik som ønsker å varsle kommunen om en eventuell svartelistet planteart. Den andre nettløsningen omhandler å behandle disse henvendelsene, og registrere og kartlegge de mulige svartelistede planteartene. Begge løsningene er implementert i Golang og lagret og strukturert på en relasjonsdatabase. Resultatet av prosjektet er en kommersiell løsning som kan bli tatt i bruk av alle kommuner i Norge.

Summary of Bachelor Project

Title:	Svartelistede Arter i Vegkanten
Date:	22.05.2023
Authors:	Malene Lundemo Anders Lunde Hagen
Supervisor:	Peter Nussbaum
Employer:	Gjøvik kommune
Contact Person:	Ingun Revhaug
Keywords:	Programmering, Kommune, Web, Relasjonsdatabase, Scrum
Pages:	64
Attachments:	4
Availability:	Åpen

Abstract: Gjøvik municipality is responsible for maintaining the diversity of plants in nature on their own region, which includes combating blacklisted plant species. The municipality requested a solution from us that would simplify the registration and therefore the combating of blacklisted plant species. Our solution includes two website solutions that communicates with each other. One recieves inquiries from residents and visitors in Gjøvik that wants to alert the municipality about a possible blacklisted plant species. The other website solution concerns administrating these inquiries, and registering and mapping the possible blacklisted plant species. Both solutions have been implemented using Golang and stored and structured on a relational database. The result of the project is a commercial solution which can be utilized by all municipalities in Norway.

Forord

Vi ønsker å takke alle involvert i denne bacheloroppgaven. Vi vil takke veilederen vår, Peter Nussbaum, for et godt samarbeid og kontinuerlig veiledning under prosjektet. Vi vil takke Gjøvik kommune og oppdragsgiveren vår, Ingun Revhaug, for et engasjert og hjelpsomt samarbeid gjennom utviklingen av løsningen vår. Til slutt takker vi hverandre for en lærerik og vel gjennomført bacheloroppgave.

Innhold

Forord . . . . . **iii**
Innhold . . . . . **iv**
Figurer . . . . . **vii**
Tabeller . . . . . **ix**
Akronymer . . . . . **x**
Ordliste . . . . . **xi**
1 Introduksjon **1**
1.1 Bakgrunn 1
1.2 Målgruppe 2
1.3 Oppgavebeskrivelse 2
1.4 Rammer 4
1.5 Avgrensning 4
1.6 Valget av oppgaven 5
1.7 Gruppemedlemmene 6
1.8 Roller 7
1.9 Rapportstruktur 8
2 Kravspesifikasjon **9**
2.1 Use Cases 9
2.2 Domenemodell 11
3 Utviklingsprosess **12**
3.1 Utviklingsmetodikk 12
3.2 Gjennomføring av metodikk 13
3.2.1 Oppgavefordeling 13
3.2.2 Sprintmøter 14
3.2.3 Git 15
3.2.4 Møtereferat og timelogg 16
4 Design og arkitektur **17**
4.1 Brukerskjema 17
4.2 App eller nettside? 24
4.3 Admin 25
5 Implementering **33**
5.1 Backend 33
5.1.1 Application Programming Interface 33
5.1.2 Brukerskjema 34

5.1.3 Admin . . . 35
5.2 Frontend . . . 37
5.2.1 Brukerskjema . . . 37
5.2.2 Admin . . . 37
5.3 Database . . . 39
5.3.1 MariaDB på Ubuntu . . . 39
5.3.2 MariaDB med Golang . . . 41
5.3.3 Synkronisering . . . 45
6 Teknologier . . . 46
6.1 Golang . . . 46
6.2 Golang HTML/Template . . . 47
6.3 Hosting . . . 48
6.4 Docker . . . 49
6.5 Database . . . 49
6.6 Teknologiskisse . . . 50
7 Testing . . . 51
7.1 Brukertester . . . 51
7.1.1 Diskusjon: Skal brukerskjemasiden ha bildeeksempler på svartelistede planter Gjøvik er på utkikk etter? . . . 52
7.1.2 Diskusjon: Skal vi la kommunen få bruke innsendte bilder videre? . . . 52
7.1.3 Diskusjon: Skal vi kreve at en henvendelse må inneholde et bilde? . . . 53
7.1.4 Diskusjon: Gammelt vs. nytt design på brukerskjema . . . 53
7.2 Manuell testing . . . 55
8 Installasjon . . . 57
8.1 Kommersiell løsning . . . 57
8.2 Database . . . 58
8.3 Sikkerhet . . . 58
8.4 Vår implementasjon . . . 58
9 Avslutning . . . 59
9.1 Diskusjoner . . . 59
9.1.1 Fra skreddersydd løsning for Gjøvik kommune til en kommersiell løsning . . . 59
9.1.2 Effektiv databasehåndtering . . . 61
9.2 Kritikk av oppgaven . . . 61
9.2.1 Samarbeid med kommunen . . . 61
9.3 Videre arbeid . . . 62
9.3.1 Kunstig intelligens . . . 62
9.3.2 Samarbeid med andre systemer . . . 62
9.3.3 Tilgjengelighet . . . 63
9.4 Evaluering av gruppen . . . 63
9.5 Konklusjon . . . 64
Bibliografi . . . 65

A Prosjektplan **68**
A.1 Mål og rammer 69
A.1.1 Bakgrunn 69
A.1.2 Prosjektmål 70
A.1.3 Rammer 71
A.2 Omfang 72
A.2.1 Fagområde 72
A.2.2 Avgrensning 72
A.2.3 Oppgavebeskrivelse 73
A.3 Prosjektorganisering 75
A.3.1 Ansvarsforhold og roller 75
A.3.2 Rutiner og grupperegler 75
A.4 Planlegging 76
A.4.1 Prosessrammeverk 76
A.4.2 Plan for statusmøter 77
A.5 Organisering av kvalitetssikring 77
A.5.1 Dokumentasjon, standarder og verktøy 77
A.5.2 Testing 79
A.5.3 Risikoanalyse 79
A.5.4 Tiltak for risikoer 80
A.6 Plan for gjennomføring 81
A.6.1 Gantt-skjema 81
A.6.2 Milepæler 82
B Prosjektavtale **83**
C Møtereferater **90**
D Timelogg **102**

Figurer

2.1 Use Cases med aktører og tjenester - Diagram laget ved hjelp av Draw.io 9
2.2 Domenemodell med enheter, egenskaper og relasjoner - Diagram laget ved hjelp av Microsoft Power Point 11

3.1 Git-issue med Git-tasks som deler en stor oppgave fra 'Backlog' . . . 15

4.1 PC-versjonen av det første brukerskjemaet på én side 17
4.2 Mobilversjonen av det første brukerskjemaet på én side 18
4.3 PC-versjonen av side nummer én, på det nye brukerskjemaet 19
4.4 Mobilversjonen av side nummer én, på det nye brukerskjemaet . . . 19
4.5 Et nytt felt dukker opp når man velger 'annet' i listen av planter . . 20
4.6 PC-versjonen av side nummer to, på det nye brukerskjemaet 20
4.7 Mobilversjonen av side nummer to, på det nye brukerskjemaet . . . 21
4.8 Eksempel på at posisjonen ble funnet fra et bilde, i dette eksempelet brukte vi et bilde tatt av 'London Bridge' 21
4.9 PC-versjonen av side nummer tre, på det nye brukerskjemaet 22
4.10 Mobilversjonen av side nummer tre, på det nye brukerskjemaet . . 23
4.11 Et eksempel på forhåndsvisning av to bilder som er lastet opp av brukeren på mobilversjonen 23
4.12 Dersom noe ikke er fylt ut riktig i brukerskjemaet, markeres det i rødt 24
4.13 Faner og sorteringsknapp for henvendelser 25
4.14 Boksen som dukker opp når man klikker på 'Sorter & filtrer' knappen 26
4.15 Nettsiden for administrering av én enkel henvendelse 26
4.16 Advarsel om endringer som ikke er lagret 27
4.17 Menyen hvor man kan navigere mellom nettsidene, inkludert begge versjonene av brukerskjemaet 28
4.18 Et eksempel på en markør av en offentlig henvendelse på det offentlige kartet 28
4.19 Egne markører for 'kart' nettsiden 29
4.20 Eksempel på filtrering av kartmarkører 30
4.21 Mulig å velge flere henvendelser samtidig for å opprette en PDF . . 31

4.22 Man kan legge til en kommentar på hver henvendelse før PDFen opprettes 32
4.23 En funksjon i Javascript som oppretter en ny Cookie 32

5.1 APIer for nettsidene 33
5.2 APIer for håndtering av data 34
5.3 APIer for statiske filer 34
5.4 Sjekker størrelse på henvendelsen før den blir tatt med videre . . . 35
5.5 Henvendelse: Før og etter redigering av navn og beskrivelse 36
5.6 Dashboard: Legge til en ny plante, smørblomst 36
5.7 Brukerskjema: Før og etter 'smørblomst' er lagt inn 36
5.8 En for-loop som går igjennom alle henvendelser med 'HTML/template' pakken fra Golang 38
5.9 En if-setning som enten legger til teksten "Ingen bildereller BBilder:basert på om det er vedlagt bilder med henvendelsen eller ikke 38
5.10 Dataene som ble endret, endres først i Javascript, dermed sendes dataene inn til APIen på serveren 39
5.11 Tabell 'PlantInquiry' slik den er satt opp på databasen 'svartelisteDB' 40
5.12 Tabell 'PlantComment' slik den er satt opp på databasen 'svartelisteDB' 41
5.13 Go-kode som lager en database handle for å kommunisere med MariaDB serveren vår 41
5.14 Go-kode som henter én henvendelse fra databasen 42
5.15 Go-kode som henter alle kommentarene knyttet til en henvendelse fra databasen 43
5.16 Go-kode som lagrer én ny henvendelse i databasen 44
5.17 Go-kode som oppdaterer én henvendelse i databasen med status 'processing' 44

6.1 Ytelse til ulike 'template engines', hvorav 'HTML' er 'HTML/template' pakken vi brukte 47
6.2 Advarsel som gis når man kobler til localhost med HTTPS 48
6.3 Teknologiskisse med visuelle fremstillinger av teknologiene vi har brukt - Diagram laget ved hjelp av Draw.io 50

7.1 PC- og mobilversjon av side nummer to, på det nye brukerskjemaet 54
7.2 En del av koden for oppretting av en test henvendelse 56

A.1 Gantt-skjema 81

Tabeller

A.1 Navn på verktøy og beskrivelse 78
A.2 Risikoanalyse 79
A.3 Tiltak på risikoer 80

Akronymer

API Application Programming Interface. viii, 7, 33–35, 38, 39, 46, 55

CRUD CREATE, READ, UPDATE og DELETE. 39, 41

CSS Cascading Style Sheets. xi, 24, 34, 47

HTML HyperText Markup Language. viii, xi, 34, 37, 38, 46, 47

HTTP Hypertext Transfer Protocol. 21, 48

HTTPS Hypertext Transfer Protocol Secure. viii, 21, 33, 48

JS Javascript. xi

PDF Portable Document Format. vii, viii, 10, 30–32, 59, 60, 62

SSL Secure Sockets Layer. 48

VPN Virtual private network. 48, 58

XP Extreme Programming. 12

Ordliste

Application Programming Interface Et grensesnitt som tillater et program eller programvare å endre et annet program eller programvare, slik at man for eksempel kan endre noe i databasen ved at brukeren trykker på en 'lagre' knapp i frontend.. x

backend Innenfor programmering refererer backend til den delen av programvaren som er bak kulissene og sluttbrukeren ikke vanligvis interagerer med, som f. eks. at en forespørsel sendes via nettet når en bruker trykker på en knapp.. 5–7, 35, 37, 47

container En pakke med programvare som innholder alle de nødvendige elementene den trenger for å kjøre i ethvert miljø. En container virtualiserer operativsystemet og kan kjøres hvor som helst. 49, 57

containerize Kjøre en applikasjon i et miljø som er satt opp i et spesifikt operativsystem og tildelt kun viktige ressurser. 49

Dockerfil Et tekstdokument som inneholder alle kommandoene i rekkefølge som trengs for å bygge en container image, i.e., kildekoden til et Docker image. 49, 57

Dockerized Prosessen av å konvertere en applikasjon til å kjøres i en Docker Container og lage Dockerfilen for den. 57

fremmednøkkel En henvisning til andre data i en database. Fremmednøkkelen peker vanligvis til en primærnøkkel i en annen tabell. 40

frontend Innenfor programmering refererer frontend til den delen av programvaren som sluttbrukeren interagerer med, som f. eks. HTML, CSS og JS filene på en nettside.. xi, 5, 7, 13, 34, 35, 46, 47, 55, 58

IaaS Infrastructure as a Service, eller infrastruktur som en tjeneste, hvor man kan bygge en virtualisert infrastruktur via nettet. 39, 48

merge Brukt i sammenheng med Git repoer hvor man tar ett repo og slår sammen med et annet repo. 15, 16

NULL Det totale fraværet av en verdi, i.e., ukjent. 40

Open Source Distribuert programvare som har kildekoden tilgjengelig for brukerne. 39, 49, 50

primærnøkkel Et felt i tabellen fra databasen som garanterer at hver rad får en unik ID. 40

repo Et Git repository, eller repo, er en måte å skille ulike versjoner av programvaren, slik at man kan jobbe usynkront før man til slutt slår dem sammen. xi, 14, 39

root En superbruker som er autorisert til å utføre sikkerhetsrelevante funksjoner som vanlige brukere ikke er autorisert til å utføre. 39

scrum Et rammeverk brukes blant annet innenfor programvareutvikling, hvor man i en gruppe har jevnlige sprintmøter hvor man bryter ned målet til mindre mål. 6, 7, 12–15, 63

SQL-injeksjon En angrepsmetode mot datasystemer der SQL-setninger blir manipulert til å gjøre ting de ikke er ment å gjøre. 41, 43

struct En samling av flere datapunkt på inn i ett datapunkt som brukes i Golang.. 42

thumbnail Et miniatyrbilde som forhåndsviser en video eller et større bilde.. 11, 40, 42, 43

Kapittel 1

Introduksjon

1.1 Bakgrunn

Verden består av mange forskjellige plantearter spredt rundt omkring hele jordkloden [1][2]. Noen arter lever i symbiose, mens andre arter er mer selvstendige. For å overleve, har noen planter utviklet seg ekstra høye for å skaffe seg mer sollys, mens andre planter har gjort seg ekstra fargerike for å sørge for at pollen blir spredt ved hjelp av insekter. Det er en konstant kamp mellom plantene om hvem som overlever, og noen ganger så blir en plante så utkonkurrert at de ikke lenger klarer å overleve eller spre seg, som gjør at plantearten til slutt dør ut.
Noen steder kan det være mange planter, som gjør at plantene må være flinke til å spre seg fort, gjøre seg stor eller overleve på lite næring. Dersom disse plantene sprer seg til et sted hvor det ikke er mye motstand fra andre planter, kan det være at de tar helt over og utkonkurrerer andre plantearter. Dette kan skje naturlig, men det kan også skje som følge av globalisering [3]. Dersom en plante blir introdusert i et nytt område, kan det være at den blir svartelistet i det området.

I Norge har vi en rekke lister over diverse planter og arter på artsdatabanken.no. Her er det blant annet en liste over svartelistede arter som utgjør en risiko for norsk natur. Svartelistede arter er et eldre begrep som har blitt endret til 'fremmed arter' [4], men ettersom oppgavetittelen inneholder ordet svartelistedeså vil vi også bruke dette begrepet fremover.
Et eksempel på en svartelistet art er kjempebjørnekjeksen. Opprinnelig fra de vestlige delene av Russland, men ble introdusert til andre deler i Europa av mennesker på 1800-tallet [5][6]. Kjempebjørnekjeksen er et godt eksempel på en plante som andre planter i Norge ikke har lært seg å konkurrere imot. Den er stor og høy, med store blader, som skygger solen for andre planter. Den sprer seg lett, da hver plante produserer titusenvis av frø. I tillegg er den giftig for mennesker, da plantesaften kan gi utslett på huden. Ansvaret i Norge ligger på et kommunalt nivå. Dette betyr at Gjøvik kommune er ansvarlig for å opprettholde naturmangfoldet, som inkluderer bekjemping av svartelistede plantearter, heretter referert som svartelistede arter.

1.2 Målgruppe

Oppgaven består av to deler, brukerskjemaet og administrasjonssidene. Målgruppen til brukerskjemaet kan bestå av alle som er innbyggere i Gjøvik kommune, med andre ord må brukerskjemaet være tilgjengelig for alle personer i alle aldre. Selv om skjemaet er tilgjengelig for alle, så lener målgruppen mot den eldre generasjonen. Oppgaven handler om svartelistede arter, som den yngre generasjonen kanskje ikke er like vant med, siden de ofte har erfaring med hagearbeid og planteliv og derfor mindre interesse. Kunnskapen om svartelistede planter er ofte høyere hos eldre fordi de har en egen hage hvor de holder på med hagestell og lignende, og derfor får mer innblikk i hvilke planter som er ønskelige. Dermed er det viktig å utvikle en nettside som er enkel å forstå og forholde seg til, ettersom det lener mot en eldre målgruppe som kan ha litt mindre erfaring når det gjelder nettbaserte løsninger.

Administrasjonssidene vil også ha en veldig utvidet målgruppe. Løsningen vil bli administrert av personer som er ansatt i en kommune, og disse kan være i alle aldre, noe som vi må ta hensyn til. Merk at denne løsningen vil da ikke være relevant for barn.

Begge løsningene må altså være tilgjengelig og forståelig for alle personer i alle aldre. Dette krever at vi må fokusere på enkelhet og unngå løsninger som krever nettbasert erfaring, siden det kan oppstå brukere som har lite erfaring.
For rapporten er målgruppen først og fremst veileder og sensorer, men også medstudenter og utviklere. Det forventes at de har en viss innsikt innen programmering, og derfor er ikke alle begreper blitt forklart i detalj. Når dette er sagt er også oppdragsgiveren en potensiell leser av oppgaven.

1.3 Oppgavebeskrivelse

Vår oppgave er å forenkle registreringen og dermed bekjempelsen av svartelistede arter i Gjøvik kommune. Dette er en trinnvis oppgave som vi mener trenger to separate løsninger som kommuniserer med hverandre.
Første trinn ligger hos en vanlig innbygger i Gjøvik som ønsker å varsle kommunen om en eventuell svartelistet art.
Andre trinn er å behandle denne henvendelsen og registrere den mulige svartelistede arten og kartlegge den.
Tredje og siste trinn er å redusere bestanden til de svartelistede artene som er kartlagt.

Løsning 1 har som mål å dekke første trinn nevnt ovenfor. Dette blir en nettside som er tilgjengelig for alle aktive innbyggerne og besøkende i Gjøvik kommune som oppdager en plante de tror er svartelistet. Nettsiden skal ha disse funksjonalitetene (merk at brukeren i punktene under kan være hvem som helst i Gjøvik som har mulighet til å besøke nettsiden):

- Nettsiden må kunne besøkes gjennom mobilen for å gjøre den lett å bruke.
- Brukeren skal kunne sende inn et bilde av planten de tror er svartelistet/-farlig, samt en liten beskrivelse, e.g., detaljer eller spørsmål.
- Brukeren kan få en tilbakemelding om at henvendelsen er tatt imot og eventuelt en til når planten er fjernet, dersom ønskelig av brukeren.
- Nettsiden må for hver henvendelse hente inn posisjonsdata om hvor planten befinner seg.

Løsning 2 skal dekke de to siste trinnene nevnt ovenfor. Henvendelser fra løsning 1 skal bli sendt til løsning 2 for behandling. Behandlingen inngår å vurdere den mulige svartelistede arten og kartlegge den. Dette vil foregå gjennom en nettside. Funksjonalitetene til denne nettsiden er som følger (merk brukeren i dette tilfellet er en ansatt i Gjøvik kommune med spesifikk tilgang):

- Brukeren må ha brukernavn og passord.
- Den skal være enkel å navigere og bruke.
- Den skal holde oversikt over alle nye og tidligere innsendte henvendelser fra aktive personer i Gjøvik kommune (se løsning 1).
- Brukeren skal kunne markere at plantearten som er henvendt er farlig eller ufarlig, i.e., registrere at dette er en svartelistet planteart.
- Registreringen i punktet over vil føre til at plantens posisjon kartlegges. Målet er at videre skal traktorførere og andre arbeidere som bekjemper plantene kunne benytte seg av samme posisjonsdata.
- Brukeren skal kunne se, endre, fjerne og legge til posisjonsdata for de svartelistede artene.
- Brukeren må ha oversikt over historikk av bekjempelsene som er gjort eller skal skje.
- Kartleggingen av de svartelistede artene skal kunne eksporteres og lagres i kommunens kartsystem.
- Det skal gjennom nettsiden kunne registreres nye plantearter i artsdatabanken.

1.4 Rammer

Oppdragsgiver hadde ikke mange rammer på starten av oppgaven, men underveis kom vi fram til at alt innhold skulle være tilgjengelig på både pc og mobil. NTNU hadde også et par rammer da dette er en bacheloroppgave med en tidsfrist.

- Vi har en begrenset tidsramme på 4 måneder, hvor oppgaven og rapporten skal være ferdig levert innen 22. mai 2023.
- Nettsiden skal fungere på alle platformer for både mobil og pc.
- Henvendelser og kartdata skal kunne lagres på servere eid av Gjøvik kommune.

1.5 Avgrensning

Løsningen vår skal kunne kommunisere med eksisterende kartløsninger som brukes i de fleste kommuner i Norge. Vi skal lage et grensesnitt som gjør at en fra kommunen kan utnytte de kartlagte svartelistede artene i sine kartsystem. Oppgaven avgrenses derfor til at vi ikke skal lage en skreddersydd løsning for Gjøvik kommune, men da et mer kommersielt fokus.
Kjennskap om svartelistede arter og statistikk om naturmangfoldet i Gjøvik er ikke vår oppgave, denne kunnskapen ligger hos Gjøvik kommune, i dette tilfellet landbrukssjefen. Alle henvendelser om svartelistede arter skal derfor kontrolleres manuelt i vår løsning, slik at kommunen selv har ansvar for at henvendelsen blir vurdert korrekt.

I tillegg vil vi ikke prøve å koble vårt system opp mot andre sine systemer, som for eksempel automatisere at henvendelser blir sendt ut til de som kjører traktorer og klipper veikantene, da det ikke finnes noen standardiserte systemer i dag som vi kan koble oss opp mot. Vi vil dermed fokusere mer på en intern løsning når det gjelder å dele henvendelser til de som skal bekjempe plantene.

1.6 Valget av oppgaven

Før valget ble tatt, hadde begge gruppemedlemmene en stor interesse av denne oppgaven. Den passet godt overens med tidligere emner vi har hatt i bachelorstudiet vårt, samtidig som at den bød på en solid utfordring basert på det vi allerede har lært.

Dagen det var presentasjoner av oppgavene, var oppdragsgiver der for å presentere oppgaven. Allerede den dagen fikk vi et godt innblikk i hva oppgaven handlet om, og at oppdragsgiver faktisk hadde et behov for en løsning på problemet sitt. Som studenter, er vi vant med å løse oppgaver som ofte ikke har nytte annet enn å lære oss programmering. Med denne oppgaven kunne vi få et større innblikk i hva man faktisk gjør ute i jobb, hvor man må finne en løsning samtidig som man tilpasser seg det virkelige liv.

Denne oppgaven passet også veldig godt i forhold til hva gruppemedlemmene ønsket i en oppgave. Den ville kreve en del frontend, og backend med server og database. Her er gruppemedlemmene såpass ulike at det den ene gruppemedlemmene foretrekker å jobbe med, ønsker ikke det andre gruppemedlemmet å jobbe med. Dermed var det nesten helt perfekt at begge gruppemedlemmene kunne jobbe med sine foretrukne områder, og til sammen skape et ferdig produkt.

Begge gruppemedlemmene har vært glade for valget av oppgaven under hele prosjektet. Den har gitt et veldig godt perspektiv på hvordan man må gå fram for å løse et problem. Det hender at den beste tekniske løsningen ikke samsvarer med det brukeren ønsker, som gjør at man må tenke på nye måter for å finne andre løsninger. Vi har også sett at det ikke bare er Gjøvik kommune som ønsker å finne en løsning på dette problemet, da andre kommuner i Norge sliter med svartelistede arter. Dette har gjort at lignende løsninger har vært vanskelig å finne, og vi har måttet funnet våre egne løsninger sammen med oppdragsgiver og andre fra kommunen.

1.7 Gruppemedlemmene

Begge gruppemedlemmene går bachelor i programmering, som har gitt oss en rekke forskjellige kompetanser fra ulike emner. Alle emner har økt kompetansen vår innenfor programmering, men for å nevne noen som er ekstra relevant for denne oppgaven, har vi hatt emner som har lært oss skyteknologier og bruk av Golang, fra 'PROG2005 - Cloud Technologies' [7] og 'PROG2006 Avansert Programmering' [8]. Vi har også hatt emner som har lært oss bruk og integrering av databaser fra emnet 'IDATG2204 - Datamodellering og databasesystemer' [9]. I tillegg har vi hatt emner innenfor design og bruk av Javascript og CSS i samsvar med backend, ved hjelp av 'PROG2053 - Webteknologier' [10].

Nylig hadde vi også et emne 'PROG2052 - Integrasjonsprosjekt' [11], som kan sees på som en 'mini-bacheloroppgave'. Her fikk vi god innsikt i å jobbe med et prosjekt over en lengre periode med en gruppe, samtidig som man bruker ulike verktøy som Scrum og Git for å forbedre arbeidsflyten. Dette emnet var til god hjelp for å lære oss overgangen fra skolerelaterte oppgaver som ofte er korte og presise, til en prosjekttankegang hvor man må tolke oppgaven litt mer og tilpasse løsningen i forhold til problemet.

I tillegg til vårt nåværende bachelorstudium, har gruppemedlemmene fått ekstra kompetanse andre steder. Anders gikk blant annet ett år på Universitetet i Oslo hvor han studerte interaksjonsdesign, hvor han plukket opp en del kompetanse innenfor objektorientert programmering og design av nettbaserte løsninger, hvor universelt design og viktige designprinsipper sto sentralt.

Malene har et års erfaring som utvikler på firmaet VisBook som tilbyr et PMS-system for hotell [12]. Der lærte hun å videreutvikle et allerede stort program etter kundens behov og sjefens visjon, samt arbeide i team. Hun har også et årsstudium i informatikk med valgemner som hadde fokus på IT-sikkerhet. Fra dette årsstudiet lærte hun hva IT-sikkerhet innebar, e.g., hvordan måle og vurdere risiko i IT, og hvilke tiltak man kan gjøre for å senke en risiko.

1.8 Roller

Vi skiller ofte mellom brukerskjema og administrasjonssidene, men i forhold til roller så ser vi heller på frontend og backend til disse sidene kombinert.
Anders Lunde Hagen har fått hovedansvaret for frontend, og har i tillegg ansvaret for Google kontoen som sørger for at vi kan bruke Google APIer til bruk av kartet til Google.
Malene Lundemo er Scrummaster og har hovedansvaret for database, Docker og innlogging. Hvis gruppemedlemmene skulle bli uenige om noe, har Malene siste ord i saken som Scrum master.
Backend er stort sett delt da det er relevant for både frontend og databasen.

Ettersom det bare er to stykker som jobber med oppgaven, så blir det naturlig at begge jobber utenfor hovedansvaret sitt. Hovedansvaret er delt opp etter personlige meninger og litt ekstra erfaring på de områdene, men igjennom prosjektperioden har gruppemedlemmene jobbet tett sammen på alle områder. I tillegg at det bare er to stykker, så er det mange deler av oppgaven som hører godt sammen, som for eksempel at databasen blir vanskelig å jobbe med hvis frontend ikke sender riktige data.

Øvrige roller består av oppdragsgiver og veileder. Veilederen vår er Peter Nussbaum, som vi har hatt fast møte med hver onsdag igjennom prosjektperioden. Han har som ansvar å hjelpe gruppemedlemmene å holde fokus på oppgaven og dirigere oss i riktig retning. Han leser også igjennom utkast av prosjektplan og prosjektrapport for å hjelpe oss der vi er usikre og trenger svar.
Oppdragsgiveren vår er Ingun Revhaug fra Gjøvik kommune, som vi har hatt møte med cirka annenhver uke, noen ganger oftere ettersom hvilket behov vi har hatt for tilbakemeldinger og innspill. Ingun sin rolle har først og fremst vært og gi oss oppgaven vi skal lage, men hun har også igjennom prosjektperioden vært med på å justere oppgaven imens vi utvikler den for å gjøre den best mulig for Gjøvik kommune.

1.9 Rapportstruktur

Rapporten er skrevet på norsk, men vi har brukt noen engelske ord som ikke er oversatt til norsk. Dette er fordi ikke alle gloser har en god norsk oversettelse, og derfor er låneordene brukt i teksten slik de skrives på engelsk. De fleste av disse glosene har derimot en norsk forklaring funnet i 'forkortelser' og 'ordliste'. Utvikling er et internasjonalt interesseområde som bruker engelsk, og vi har derfor forholdt oss til å bruke engelsk i koden vår. Vi nevner også at oppdragsgiveren vår krevde at vi skulle bruke ordet 'bekjempelse' for å beskrive prosessen av å fjerne svartelistede planterarter, og dette er brukt flere steder i rapporten.

Rapporten er delt inn i 9 kapitler:

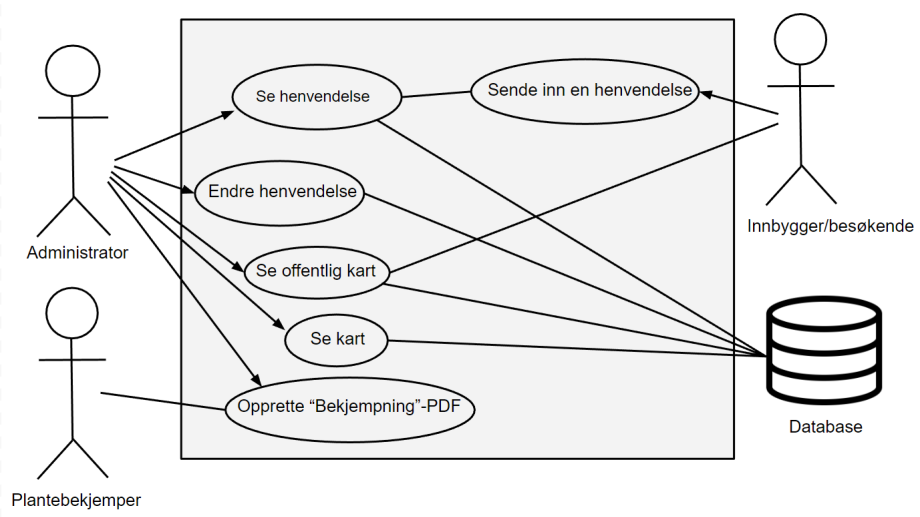
1. **Introduksjon**, introduserer oppgaven med tilhørende teori og målsetning, i tillegg til struktur på rapporten.
2. **Kravspesifikasjon**, som inneholder en rekke diagrammer og krav som angår oppgaven
3. **Utviklingsprosess**, viser til planlegging og ukentlige møter i løpet av prosjektet
4. **Design og arkitektur**, som går over ulike designvalg som ble tatt og hvorfor vi tok de valgene
5. **Implementering**, som går over koden i detalj og hvordan vi har implementert ulike aspekter fra punkt 4.
6. **Teknologier**, som diskuterer hvilke teknologier vi har valgt og hvorfor vi valgte dem over andre teknologier.
7. **Testing**, som viser ulike tester vi har gjennomført.
8. **Installasjon**, som viser hvordan oppgaven kan installeres andre steder.
9. **Avslutning**, diskusjon og konklusjon på hvordan oppgaven har gått og eventuelle endringer vi kunne gjort i fremtiden.

Kapittel 2

Kravspesifikasjon

2.1 Use Cases

Kravspesifikasjon er en måte å finne ut *hva* man skal utvikle, og i dette kapitlet starter vi med å se på de funksjonelle kravene. Et Use Case diagram viser de funksjonelle kravene i et prosjekt. Figurene er aktører og boblene blir tjenester, i.e., handlingssekvenser, som programmet utfører. En pil fra aktør til tjeneste tilsvarer at aktøren initierer dette Use Caset, mens andre involverte aktører er vist med en strek. I figur 2.1 er vårt Use Case diagram.



Figur 2.1: Use Cases med aktører og tjenester - Diagram laget ved hjelp av Draw.io

Basert på figur 2.1 viser boblene de forskjellige tjenestene, i.e., Use Case-ene, som vår løsning utfører. Disse er beskrevet nedenfor.

Tittel: Sende inn henvendelse.
Aktør: Innbygger/besøkende

Mål: Melde om en mulig svartelistet planteart.
Beskrivelse: Innbygger/besøkende i kommunen fyller ut et nettskjema med bilde, tittel, og beskrivelse av plantearten de tror er svartelistet.

Tittel: Se henvendelse.
Aktør: Administrator, Database
Mål: Vurdere en henvendelse.
Beskrivelse: Administrator åpner en henvendelse gjennom administrasjonssidene ved å klikke på en av henvendelsene fra listen. Henvendelsen er lagret på databasen.

Tittel: Endre henvendelse.
Aktør: Administrator, Database
Mål: Behandle en henvendelsen.
Beskrivelse: Administrator endrer en henvendelse gjennom administrasjonssidene ved å endre tittel, status, eller å legge til kommentarer. Henvendelsen blir så oppdatert på databasen.

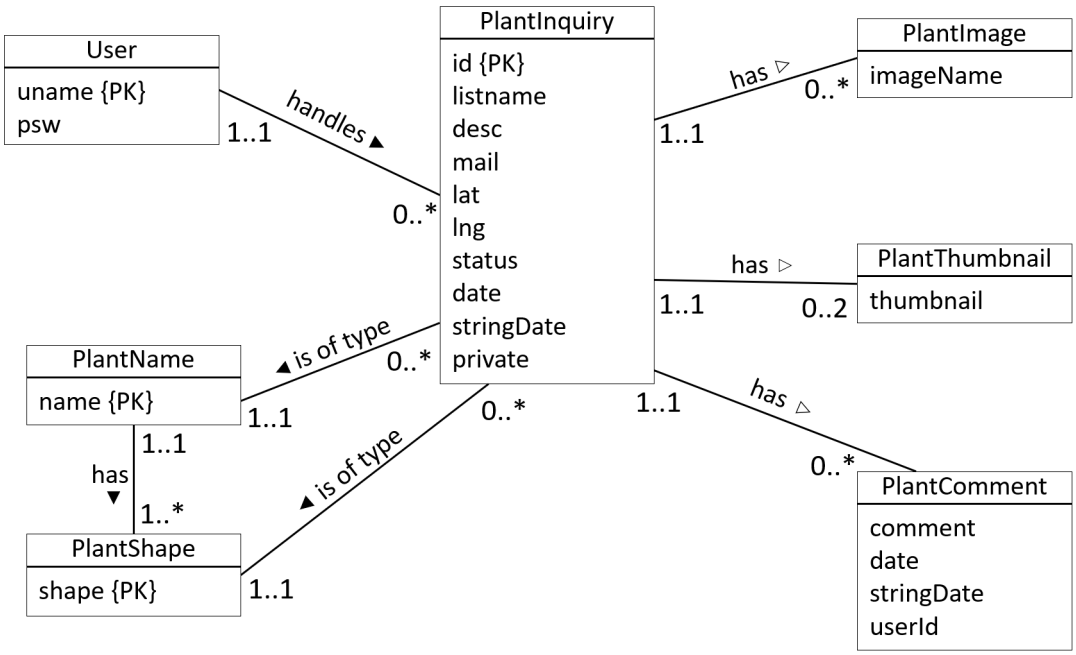
Tittel: Opprette BekjempningPDF.
Aktør: Administrator, Plantebekjemper
Mål: Opprette en PDF som kan benyttes av de som skal bekjempe planten/plantene.
Beskrivelse: Administrator trykker på Til bekjempning"og det opprettes en PDF på henvendelsen med muligheten til å legge inn kommentarer. PDF-en er ment å benyttes av plantebekjemperen. Brukeren kan også opprette en PDF med flere planter ved å markere flere henvendelser.

Tittel: Se kart.
Aktør: Administrator, Database
Mål: Få oversikt over posisjonene til alle de svartelistede artene.
Beskrivelse: Administrator trykker på Kartnettsiden og får tilgang til et navigerbart Google-kart. Kartet viser alle planteposisjonene som er hentet fra henvendelsene under behandling, og er representert med figurer og farger basert på behandlingsstatusen. Alt relatert data til henvendelsene ligger lagret i databasen.

Tittel: Se offentlig kart.
Aktør: Administrator, Innbygger/besøkende, Database
Mål: Se/referere til et kart for offentlig bruk.
Beskrivelse: Administratoren trykker på Kart (offentlig)nettsiden og får tilgang til et navigerbart Google-kart, som kan vises til innbyggere/besøkende. Dette kartet er et offentlig alternativ til tjenesten 'Se kart'. Tjenestene har de samme egenskapene, men dette kartet viser bare de henvendelsene som er offentlige, i.e., de som ikke er markert privat. Alt relatert data til henvendelsene ligger lagret i databasen.

2.2 Domenemodell

En domenemodell illustrerer de forskjellige partene i et program og hvordan de teknisk henger sammen. Figur 2.2 viser vår dommenemodell. Merk at denne modellen blir også et abstrakt bilde på hvordan databasen vår bygd opp, i.e., hvilke tabeller vi har, innholdet i dem, og sammenhengen mellom dem.



Figur 2.2: Domenemodell med enheter, egenskaper og relasjoner - Diagram laget ved hjelp av Microsoft Power Point

En henvendelse som er sendt inn av en innbygger eller besøkende i Gjøvik blir midtpunktet som knytter majoriteten av partene i prosjektet sammen. Alle henvendelser blir behandlet av én bruker, brukeren blir enheten 'User' i figuren og har egenskapene brukernavn og passord. En henvendelse er i denne figuren kalt 'PlantInquiry', og egenskapene til denne enheten er hva vi lagrer av henvendelsen. Én henvendelse kan ha flere eventuelle bilder eller thumbnails, som vist med 'PlantImage' og 'PlantThumbnail' relasjonene. Én henvendelse er av én bestemt type plante som blir representert med én type figur på kartet, i.e., relatert til 'PlantName' og 'PlantShape'. Merk at en type plante kan bli representert med samme type figur på kartet. En henvendelse kan også ha kommentarer knyttet til seg, i.e., kan en bruker av administrasjonssidene legge på flere kommentarer underveis i behandlingstiden for en henvendelse. Disse kommentarene blir kalt 'PlantComment' i denne figuren, og egenskapene viser hva vi lagrer når en kommentar opprettes.

Kapittel 3

Utviklingsprosess

3.1 Utviklingsmetodikk

Vi har valgt Scrum [13] som utviklingsmetodikk for oppgaven vår. Forrige semester hadde vi emnet PROG2052 som skulle forberede oss til bacheloroppgaven, og et av kravene i det emnet var å bruke Scrum. Dermed har begge gruppemedlemmene Scrum ferskt i minne, som gjorde det til et godt valg for oss. Scrum gir en god måte å fordele arbeid på, i tillegg til å sikre jevn flyt i arbeid med ukentlige sprinter.

Selv om vi nylig har lært om Scrum, så betyr ikke det at Scrum er det beste for prosjektet vårt. Vi tittet på andre utviklingsmetodikker før vi valgte Scrum, for å se hva de andre utviklingsmetodikkene har å tilby.

XP eller 'extreme programming' [14] er en utviklingsmetodikk som fokuserer på korte perioder med programmering. Bacheloroppgaven varer bare i 4 måneder, så det er viktig for oss å fokusere på helheten av oppgaven. Med XP så skal man bruke all tiden sin en del av prosjektet for å gjøre det ferdig, men siden vi har såpass lite tid til å gjøre hele bacheloroppgaven er det viktig for oss å bruke tiden effektivt. Vi vil dermed ha et større behov for langtidsplanlegging og mer fleksibilitet på hva vi jobber med. Dermed kan de korte periodene med fokus på noen få ting gjøre at vi får for dårlig tid eller jobber for mye på noen deler av oppgaven. XP er dermed ikke en ideell kandidat for oss.

Kanban [15] er veldig likt Scrum, men har mer fokus på å gjøre ferdig oppgaver som er i gang før man begynner på nye oppgaver. Selv om det vil være viktig for oss å gjøre ferdig oppgaver før vi fortsetter, vil ikke dette være et stort problem for oss. Det er bare to gruppemedlemmer, så det vil være enkelt å holde styr på hva som er gjort og hva som ikke er gjort. I tillegg er dette en bacheloroppgave med begrenset utviklingstid, som gjør at vi må noen ganger ta et valg om hvilke oppgaver som skal prioriteres og hvilke oppgaver som må forbli ikke fullført. Dermed kan Kanban stride litt imot arbeidsflyten vi ønsker å oppnå.

Agile [16] er på mange måter utviklet fra Scrum, som gjør det til et godt alternativ for Scrum. Det er fokus på periodevis jobbing hvor man designer, utvikler, tester, og evaluerer alt i samme periode. Periodene er alt fra én til fire uker lange og fokuset er å grundig gå igjennom hver oppgave, slik at eventuelle feil og mangler blir oppdaget tidlig. Denne måten å jobbe på er ikke veldig ulikt Scrum, som er noe vi ser etter. Hovedgrunnen til at vi ikke gikk for Agile, er at med Agile skal man ha resultatet klart på slutten av prosjektet, mens Scrum har mer fokus på ukentlige resultater.

Så selv om Agile og Scrum er veldig likt og deler mange egenskaper, er Scrum noe mer spesifikk når det kommer til hvordan man skal gå frem. I Scrum er det for eksempel roller som 'Scrum master', som gjør at man kan lettere fordele ansvar, som kan bli problematisk dersom man ikke har tydelige grenser mellom gruppemedlemmene. Agile kan ha ukentlige møter, men Scrum er litt mer tilpasset ukentlige perioder, som vi ønsker med tanke på den korte tiden vi har på bacheloroppgaven.

3.2 Gjennomføring av metodikk

Etter å ha blitt enig om å velge Scrum som metodikk, begynte vi prosessen med å fordele oppgaver, angi roller og begynne planleggingen av bacheloroppgaven.

3.2.1 Oppgavefordeling

Før vi valgte hvilken bacheloroppgave vi ville ha, var vi allerede enige om hvilke områder hver av oss foretrakk. Når vi fikk oppgaven så delte vi oppgaven i to, hvor hver av oss fikk hovedansvar for det vi foretrakk å jobbe med. Anders har både en preferanse og litt ekstra studiebakgrunn innenfor brukeropplevelsesdesign, så han tok hovedansvaret for frontend, mens Malene har noe mer erfaring innen databaseteknologier, så hun tok på seg hovedansvaret for databasen. Selve serveren vil inngå i frontend og database, så der ble ansvaret fordelt likt.

Under prosjektperioden ble oppgavene fordelt under sprintmøtene. Vi gikk gjennom backlog og valgte ut oppgaver i form av Git-issues som vi skulle arbeide med den kommende uken, basert på hvem som hadde ansvaret for hva. Dersom det ikke var like mye å gjøre på enten frontend eller database, ble andre områder dekket slik at begge to hadde like mye å gjøre hver uke. Til tross for delt hovedansvar, så var det mye kommunikasjon fram og tilbake da vi ofte trengte hjelp fra hverandre. I tillegg oppdaterte vi hverandre på nye kodesnutter dersom bare én av oss jobbet med noe, slik at begge to var oppdatert og forsto all koden.

3.2.2 Sprintmøter

I starten av bacheloroppgaven, la vi en plan på gjennomføringen av Scrum. Malene ble utnevnt Scrummaster, timeliste i Microsoft Teams ble satt opp og Git-repoet ble opprettet. Med 22,5 studiepoeng, satte vi av 30 timer i uken for arbeid med bacheloroppgaven. Vi la også en plan om å ha to sprintmøter i uken, et sprint planning møte på mandager hvor vi planlegger uken som kommer og et sprint-review møte på fredager hvor vi går igjennom hvordan uken har gått. Ett retrospektivt sprintmøte ville også bli holdt annenhver uke sammen med review møtene på fredagen. Møte med veileder ble planlagt å holdes hver onsdag, og møter med oppdragsgiver ville bli holdt ukentlig i starten, men senere i prosjektet møttes vi noe sjeldnere ettersom hvilket behov vi hadde for møter.

I løpet av bacheloroppgaven har vi hatt én uke lange sprinter, da bacheloroppgaven bare er fire måneder lang. Vi vurderte to uker lange sprinter, men siden det er bare snakk om fire måneder og vi hadde avtalt ukentlige veiledningsmøter, så var det mer naturlig å legge opp til ukentlige sprinter. Det retrospektive møtet hadde vi annenhver uke, men etter våre første to retrospektive møter i januar, følte vi at det gikk veldig bra med tanke på at vi nylig hadde hatt et emne med fokus på Scrum, så vi reduserte møtene til hver tredje uke isteden. På det første retrospektive møtet vi hadde i februar, gikk vi blant annet over sprintlengde i forhold til at vi var ute av planleggingsfasen og hadde begynt å jobbe med nettsidene, og vi kom fram til at én uke både hjalp med å holde jevn arbeidsflyt og samtidig oppdatere hverandre ofte på nye problemer som oppsto.

Møte med veileder

Våre ukentlige møter med veileder besto av å vise fram det vi hadde gjort siden sist møte, og så få veileder sine tilbakemeldinger om hvordan vi ligger an. Veileder har både gitt tilbakemelding på arbeidet vi har gjort, men har også hjulpet oss med å holde fokus på tiden og holde riktige prioriteringer.

Møte med oppdragsgiver

Møtene våre med oppdragsgiver startet med å være ukentlige. I planleggingsfasen hadde vi blant annet møter med andre fra kommunen også, som hjalp både oss og oppdragsgiver i å tilrettelegge oppgaven. Ettersom oppgaven var ganske åpen til å begynne med, brukte vi mange møter på å planlegge hva vi skulle lage. Når planleggingen var over, så hadde vi bare to møter med oppdragsgiver i februar. Først og fremst fordi vi trengte litt tid på å sette opp nettsidene slik at vi hadde noe å vise fram, men også at det tok tid å sette sammen både brukerskjemaet og administrasjonssidene til en brukbar demo som begynte å reflektere funksjonaliteten sin.

Ettersom vi ikke hadde noen lignende løsninger å sammenligne oss med, var det viktig for oss å få fram et produkt som gjenspeilte hva oppdragsgiver ønsket. Der-

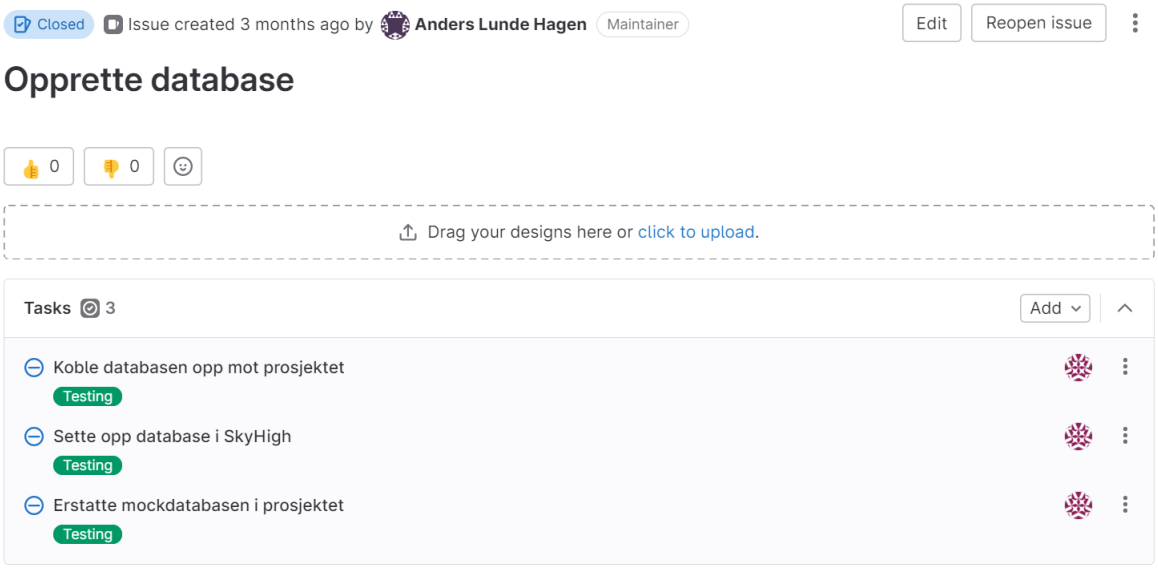
med gikk februar ut på å lage en grov nettside som var god nok til at oppdragsgiver kan legge inn sine meninger om hva som traff og ikke traff i forhold til problemet hun har. I mars var grunnstrukturen på plass og oppdragsgiver hadde fått en god ide om hva sidene går ut på, så dermed ble møtene mer fokusert på finpussing og videreutvikling av nettsidene. I april nærmet vi oss slutten og fokuset var på å gjøre nettsidene ferdige nok til at de kan gjøre alt oppdragsgiver hadde ønsker om.

3.2.3 Git

Vår Scrum-backlog var en samling av Git-issues med tittel og en beskrivelse av oppgaven. Vi brukte Git-labels for å dokumentere statusen på oppgavene. Statusene var delt i fem kategorier:

- Backlog - Oppgaver i backloggen
- Påbegynt - Oppgaver i prosessen av å bli implementert
- Testing - Oppgaver som ventet på å bli godkjent for å bli merget inn i main (hoved-branchen vår)
- Review - Oppgaver som trengte å bli gjennomgått av en annen utvikler, som for eksempel implementerte oppgaver som trenger en ekstra bedømmelse
- Bugfix - Problemer eller feil som hadde dukket opp underveis. Git-issues med dette Git-labelet ble prioritert fremfor andre

Til og begynne med var Git-issues med status 'Backlog' relativt store og måtte bli delt opp i mindre oppgaver, i.e., Git-tasks, for at oppgaven skulle bli gjennomførbar i løpet av en sprint. Et eksempel på dette er i figur 3.1. Figuren viser også at hver Git-task har sine egne Git-lables, i dette tilfelle er alle satt til testing.



Figur 3.1: Git-issue med Git-tasks som deler en stor oppgave fra 'Backlog'

Fordelen med Git var at vi kunne referere til Git-issues, og Git-tasks, når vi gjorde Git-merges. På denne måten hadde vi alltid god kontroll på hva en merge inneholdte og hva slags endringer den brakte når den ble innlemmet i main-branchen vår. I tillegg til Git-labels, og flittig bruk av Git-issues, var vi også strenge på å skrive informative Git-commits. Dette hjalp betraktelig på arbeidsflyten vår, og gjorde det mulig at vi kunne utvikle prosjektet sammen uten å ødelegge for hverandre.

3.2.4 Møtereferat og timelogg

Møtereferater ble skrevet i en egen kanal på Discord, hvor vi markerte dato, deltakere og en kort oppsummering av selve møtet. Timeloggen ble ført på et delt Excel dokument i Microsoft Teams, hvor vi har ført timer basert på hva vi har jobbet med. I timeloggen har vi delt opp timene i kategorier basert på hva vi har brukt tiden vår på.

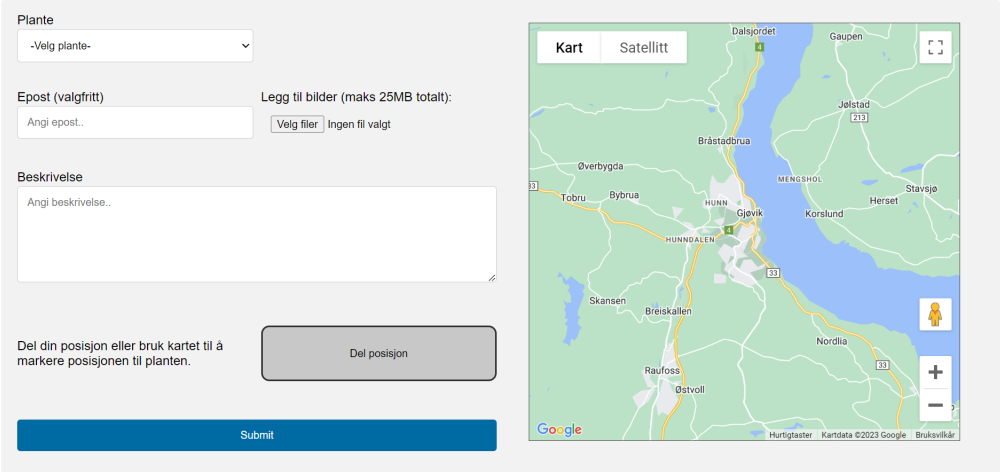
- Planlegging: Hovedsakelig for starten av bacheloroppgaven hvor vi brukte tid på planlegging og oppsett av verktøy.
- Research: Tid brukt på å finne diverse bakgrunns info på temaet svartelistede arter, bruk av verktøy og kilder til rapport.
- Dokumentasjon: Skriving av prosjektplan, rapport og andre dokumenter knyttet til rapporten.
- Koding: Tid brukt til selve kodingen.
- Testing: Tid brukt på testing av nettsidene, enten med hverandre, veileder, oppdragsgiver eller andre brukere.

Kapittel 4

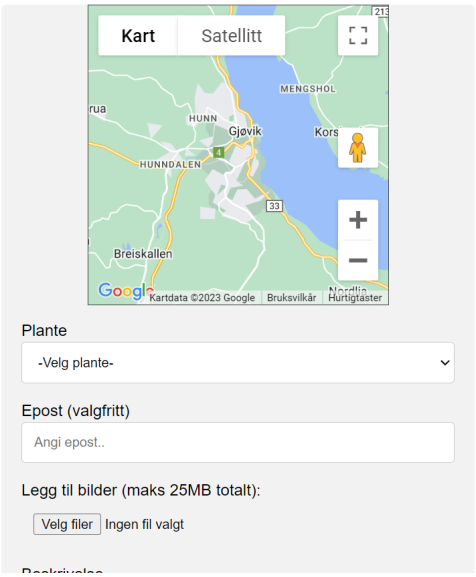
Design og arkitektur

4.1 Brukerskjema

Videre skal vi se mer på designet av nettsidene og hvilke valg vi har tatt igjennom perioden. Vi starter med vårt første brukerskjema design, hvor vi hadde lagt inn alt på én side. Det viktigste vi måtte hente fra innbyggerne i kommunen var hvilken plante det var snakk om, bilde av planten og posisjonen til planten. Alt dette kunne fint få plass på samme side, som vist i figur 4.1 og 4.2. Selve plantenavn og en eventuell beskrivelse tar ikke opp mye plass, så halvparten av siden var gitt til kartet, slik at man lettere kan navigere på kartet. På mobilversjonen er den noe mindre, men man har muligheten til å forstørre til fullskjerm for de som har behov for et større kart.



Figur 4.1: PC-versjonen av det første brukerskjemaet på én side

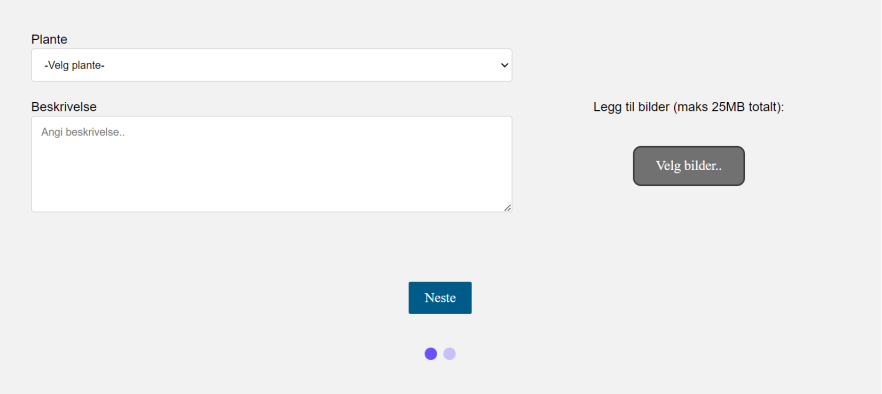


Figur 4.2: Mobilversjonen av det første brukerskjemaet på én side

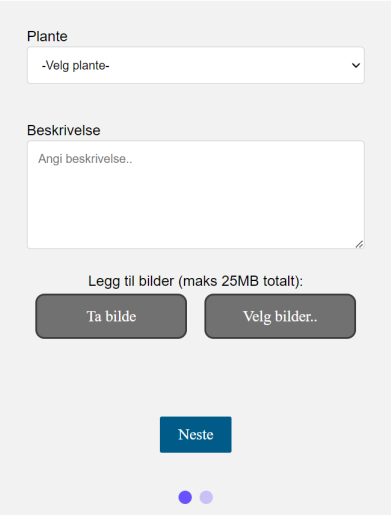
Dette designet var godt mottatt, med tanke på hvilken informasjon den prøvde å hente ut. Oppdragsgiver hadde ingenting å basere seg på før bacheloroppgaven, så det å ha et skjema hvor man kan legge inn informasjonen som man er ute etter fra en svartelistet art var godt nok. Problemet med denne løsningen kom når vi ønsket å legge til flere elementer. Vi diskuterte med veileder at vi ville ha et behov for å krysse av en godkjenning for lagring av data, som bilder og kontaktinformasjon. Siden vi allerede hadde fylt opp siden, måtte vi gjøre elementene litt mindre for å få plass til mer. Men det var begrenset hvor mye man kan redusere størrelsen på noen av elementene før det blir rotete og vanskelig å lese. Dermed begynte vi å se på en annen løsning.

Vårt nye design kopierte all funksjonalitet fra det første brukerskjemaet, men delte den opp i flere biter. Problemet med det første designet var at det var veldig mange elementer på én side, som kunne gjort det noe overveldende og forvirrende når man besøker den for første gang. Ved å dele opp hele skjemaet til flere sider, får man en mer trinnvis opplevelse når man fyller ut skjemaet. Da blir det mer som en bruksanvisning som forteller deg hva du må gjøre, som vil være en del lettere for førstegangsbrukere.

Den første siden, som vist på figur 4.3 og 4.4, ser ryddigere ut etter å ha fjernet kartet. Ved å bare fokusere på selve planten, er det lettere for brukeren å vite hva de må gjøre. Først så trenger de bare å skrive ned hvilken plante det er snakk om, og så inkludere et bilde av planten. Mest sannsynlig er man ute, rett ved siden av planten, når man tar fram brukerskjemaet, så da vil man på mobilversjonen ha muligheten til å bare ta frem kameraet med 'ta bilde' knappen og dermed slippe å bla igjennom bilder på enheten. Dette var en del enklere for brukere vi testet for, mer om dette i punkt 7.1, brukertester.

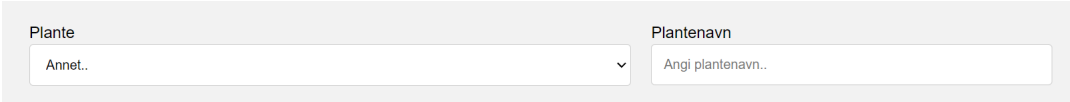


Figur 4.3: PC-versjonen av side nummer én, på det nye brukerskjemaet



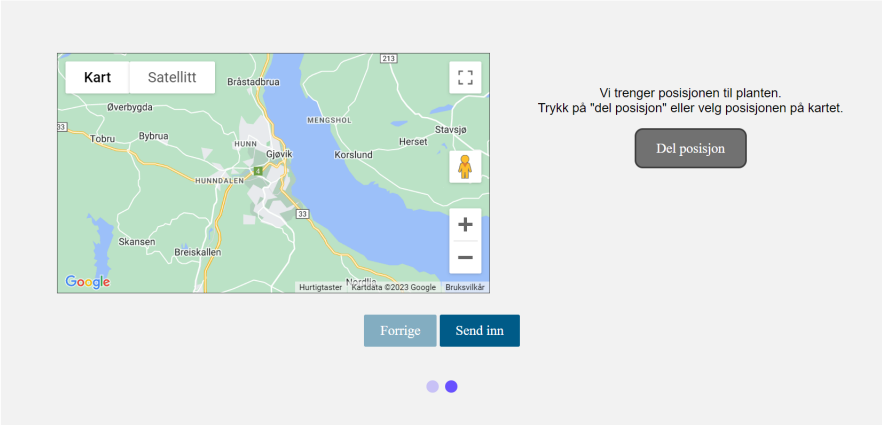
Figur 4.4: Mobilversjonen av side nummer én, på det nye brukerskjemaet

Et eksempel på å redusere mengden med informasjon gitt til brukeren, kan man se på figur 4.5 at man kan selv fylle inn et eget plantenavn, men bare dersom man klikker på 'annet' i listen over planter. Ellers vil ikke dette være relevant for brukeren, og det feltet forblir gjemt.

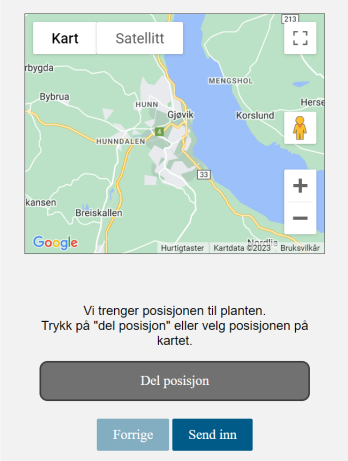


Figur 4.5: Et nytt felt dukker opp når man velger 'annet' i listen av planter

På side nummer to har vi fullt fokus på kart og posisjon, som vist i figur 4.6 og 4.7. Hovedgrunnen til at vi splittet planter og kart, var at vi integrerte en løsning på å hente ut posisjonsdata fra bilder som ble lastet opp. Når en bruker har valgt en plante og lastet opp et bilde av planten, så forsøker vi å hente ut posisjonen fra et av bildene som ble lastet opp via metadataen til bildet [17]. Hvis man finner posisjonsdata, så vil det automatisk bli lastet opp på kartet, slik at brukeren ikke trenger å dele posisjonen sin, som vist i 4.8. I det tilfellet, så kan brukeren nesten bare ignorere posisjonssteget og bare gå videre med å sende inn skjemaet.

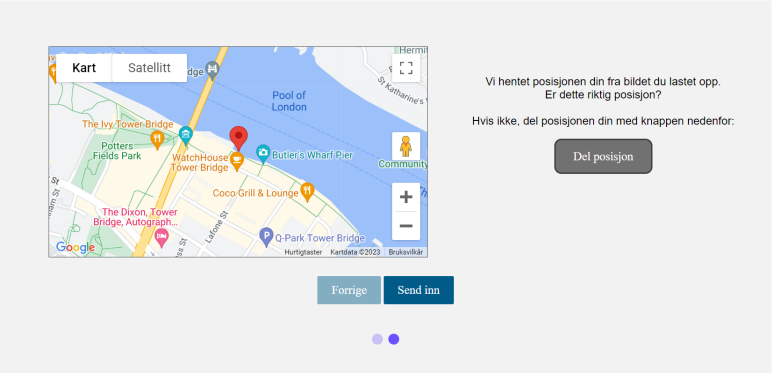


Figur 4.6: PC-versjonen av side nummer to, på det nye brukerskjemaet



Figur 4.7: Mobilversjonen av side nummer to, på det nye brukerskjemaet

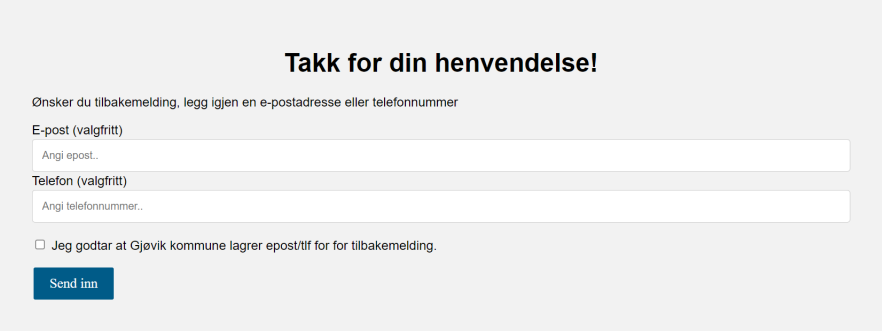
Ett av de positive sidene ved å gjøre det på denne måten, er at det ikke alltid er like lett å dele sin posisjon i en nettleser. Selv om det ofte bare dukker opp en liten melding som sier 'nettsiden ønsker din posisjon, tillatt/blokker', kan det være flere faktorer som gjør det vanskeligere, ettersom hvilke innstillinger man har på telefonen og i nettleseren. Vi fant for eksempel ut under testing at vi plutselig ikke kunne dele posisjonen vår med nettsiden lenger. Etter mye fram og tilbake, fant vi ut av at det er fordi vi kjørte nettsiden med en HTTP kobling, ikke HTTPS, og moderne nettlesere gir ikke tilgang til posisjonsdeling på HTTP sider. Selv om akkurat det problemet ikke er relevant for brukere, så ønsket vi å redusere sannsynligheten for liknende problemer ved å bruke bildene sine posisjonsdata.



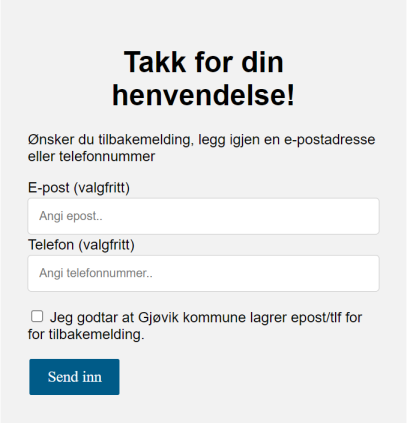
Figur 4.8: Eksempel på at posisjonen ble funnet fra et bilde, i dette eksempelet brukte vi et bilde tatt av 'London Bridge'

Det er selvfølgelig ikke bare positivt, da man fort kan bli bekymret over at nettsiden vet hvor du er uten å ha delt posisjonen din. Vi spesifiserer dermed at vi har hentet posisjonsdataene fra bildet som ble lastet opp, for og ikke hemmeliggjøre metodene våre. Ved å spesifisere at vi fikk posisjonsdataene fra bildene, kan brukeren velge å skru av denne innstillingen på telefonen, og så laste opp på nytt.

Den siste siden vises etter skjemaet har blitt sendt inn, og gir brukeren mulighet til å sende inn kontaktinformasjon. Ettersom dette ikke er informasjon Gjøvik kommune har bruk for, er dette valgfritt for brukeren. Oppdragsgiver har nevnt at det kan være noen brukere som ønsker tilbakemelding på om planten de meldte ifra om er bekjempet. For de som ikke har behov for det, så er skjemaet allerede sendt inn, som tydeliggjort med 'takk for din henvendelse'. Ettersom dette bare er relevant for noen brukere, er det en mye bedre løsning å ta det etter skjemaet er sendt inn, istedenfor å integrere det inn i selve skjemaet som det første designet gjorde. Det gir også mye mer plass for oss å skrive om hva Gjøvik kommune har tenkt til å gjøre med kontaktinformasjonen. Som vist i figur 4.9 og 4.10, så står det bare 'om du ønsker tilbakemelding', fordi vi spesifiserte aldri nøyaktig hva en tilbakemelding ville inneholdt. Denne biten var ikke veldig prioritert av hverken oss eller oppdragsgiver, da det ikke er et stort behov for tilbakemeldinger i dag. Men ettersom hele siden er dedikert til kontaktinformasjon, så er det mer enn nok plass til å legge til noen setninger om hva og hvordan kontaktinformasjonen blir brukt til, skulle det være nødvendig.



Figur 4.9: PC-versjonen av side nummer tre, på det nye brukerskjemaet

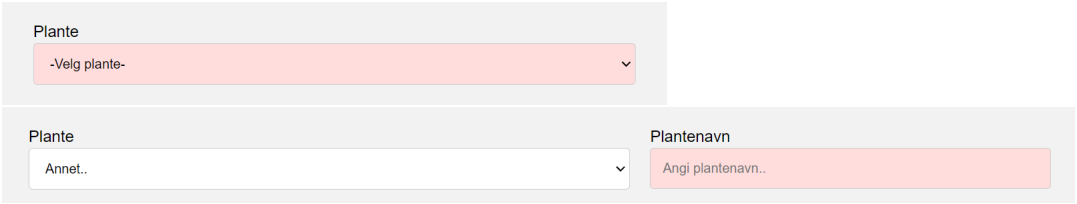


Figur 4.10: Mobilversjonen av side nummer tre, på det nye brukerskjemaet

Det er i tillegg flere mindre ting som er implementert for å gjøre opplevelsen til brukeren lettere. På den første siden, vil bildene man laster opp også dukke opp på siden, som vist på figur 4.11. Ved å vise bildene, kan man dobbeltsjekke at alle bildene er lastet opp, dersom man valgte flere bilder. I tillegg viser ikke mobilen bilde du tok, ved bruk av 'ta bilde' knappen, så da kan det være greit for brukeren å se bildet de tok i tilfelle det ble dårlig og man må ta et nytt bilde.

Figur 4.11: Et eksempel på forhåndsvisning av to bilder som er lastet opp av brukeren på mobilversjonen

Brukeren har heller ikke mulighet til å gå videre i skjemaet, dersom ikke alt er fylt ut. For eksempel må plantenavn fylles ut før man kan gå videre, som vist i figur 4.12. Det er kun plantenavn og posisjon som er krevd for å sende inn brukerskjemaet, men dette kan lett utvides til å kreve opplastning av bilde og eventuelt legge til en beskrivelse. Om det er nødvendig å kreve bilde eller ikke, er diskutert mer i punkt 7.1, om brukertester.



Figur 4.12: Dersom noe ikke er fylt ut riktig i brukerskjemaet, markeres det i rødt

4.2 App eller nettside?

Når vi fikk oppgaven, var den veldig åpen for tolkning, da det var lite som var spesifisert. En av delene som ikke var spesifisert, men nevnt i oppgavebeskrivelsen, var at vi skulle lage en app av noe slag. Dette ble diskutert de første par ukene under planleggingen, og vi kom fram til at dersom vi skal lage en app, så ville det bare blitt en app for skjemaet som fylles ut av innbyggere. Administrasjonssidene ville stort sett blitt brukt på PC-en, så den skulle ikke være en app.

Hovedgrunnen til at det ikke ble en app til slutt var at vi ønsket en enklest mulig løsning på brukerskjemaet. Det skal ta minst mulig tid for brukeren å fylle ut skjemaet. Ved å lage en nettside, så trenger brukeren bare å finne fram til skjemaet via en lenke på Gjøvik kommune sine nettsider eller lignende. Med en app, så må man laste ned appen først og så ta opp brukerskjemaet. Dette ville økt tiden en førstegangsbruker bruker på å nå skjemaet, ettersom når man først har appen trenger man ikke å installere den igjen. I tillegg ville en app krevd ekstra tid fra oss å utvikle, da vi må lage en app for både Android og iOS. En nettside må også tilpasse seg hvilken enhet som brukes, men dette er bare noe vi må forholde oss til i CSS filene.

4.3 Admin

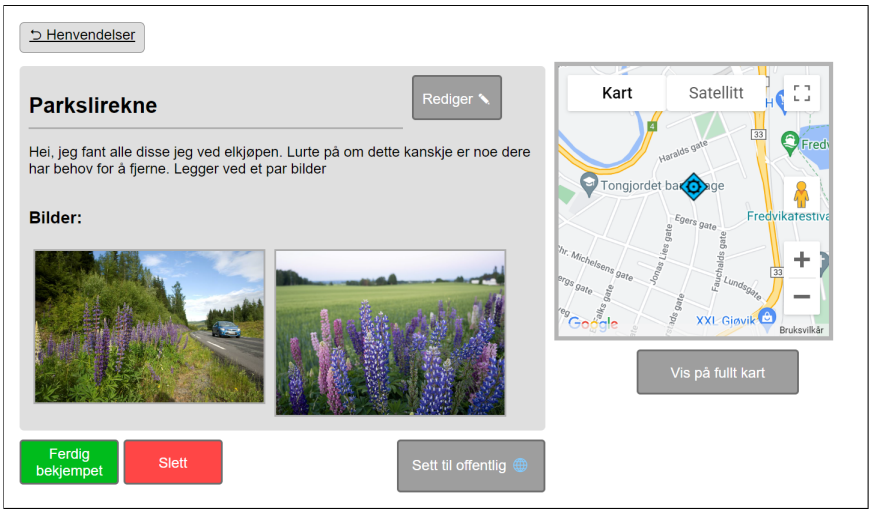
For administrasjonssidene, så er det henvendelsene som er i fokus. Henvendelsene kan ha flere statuser, som har hver sin viktighet. Nye henvendelser må godkjennes, godkjente henvendelser må bekjempes, henvendelser under bekjemping må bli ferdig bekjempet, og henvendelsene som er ferdig bekjempet må kunne bli vist fram til offentligheten. I tillegg skal det være mulig å slette henvendelser, dersom de er useriøse eller en plante som ikke er svartelistet er sendt inn.

Ettersom det etter en liten periode vil begynne å bli mange henvendelser, er det viktig å kunne både skille henvendelser med forskjellige statuser, i tillegg til å kunne sortere og filtrere på ulike måter. Som vist på figur 4.13, har vi implementert et fanesystem hvor man kan lett hoppe imellom de forskjellige henvendelsene. I tillegg er det en knapp på høyre side 'sorter & filtrer', hvor man kan sortere alfabetisk og på dato, og filtrere på spesifikke planter, som vist på figur 4.14. I fremtiden, når det har bygget seg opp nærmere hundre henvendelser, så kan det være praktisk å måtte sortere mer spesifikt enn bare alfabetisk eller på nye og gamle henvendelser. Det kunne vært nyttig å implementere en kalender hvor man kan velge en gitt datoperiode, noe som ville vært fullt mulig ettersom hver henvendelse har en dato tilknyttet til seg.

Figur 4.13: Faner og sorteringsknapp for henvendelser

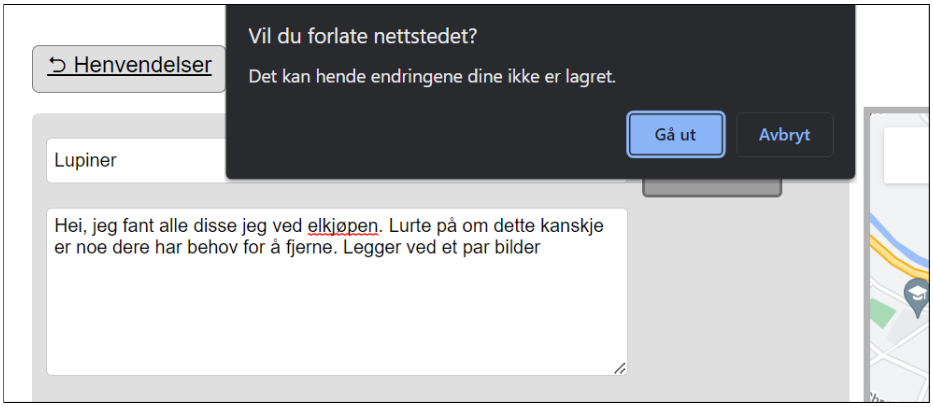
Figur 4.14: Boksen som dukker opp når man klikker på 'Sorter & filtrer' knappen

Før man klikker inn på en henvendelse for å se og administrere den, så vises den viktigste informasjonen av en henvendelse slik at man lettere kan se hvilken henvendelse det gjelder. To av bildene vises, selv om det er flere vedlagt, i tillegg til navn og beskrivelse. Antall kommentarer og dato er også tilgjengelig på denne siden. Som vist på figur 4.13, kan man se all denne informasjonen på en henvendelse om kjempebjørnekjeks.



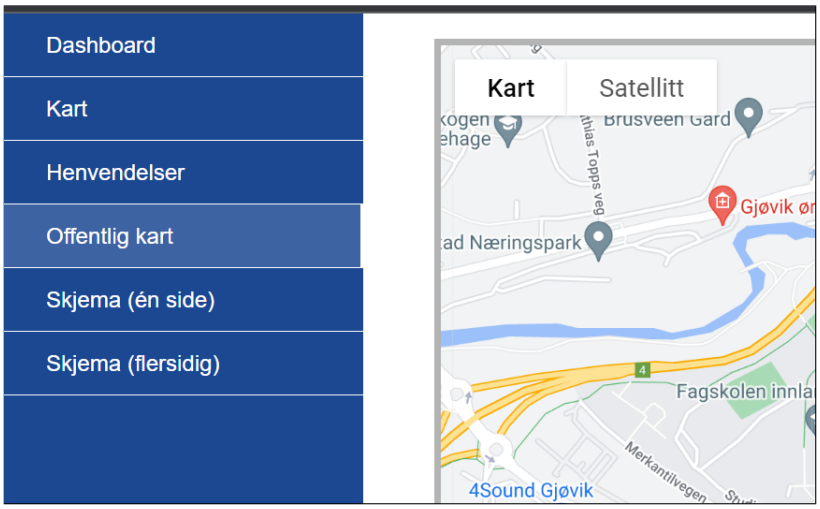
Figur 4.15: Nettsiden for administrering av én enkel henvendelse

Når man klikker inn på en henvendelse, så møter man nettsiden som er vist i figur 4.15. Hovedfunksjonaliteten til denne siden er først og fremst å kunne redigere henvendelser og endre statusen deres. Har for eksempel noen feilidentifisert en plante, men den er fortsatt svartelistet, kan man enkelt redigere navnet ved å trykke på 'rediger' knappen. Når man trykker på 'rediger', så endres teksten til inputbokser med tekst, som da kan endres på. Som vist i figur 4.16, så kan man endre teksten i tekstboksene og trykke 'lagre'. Hvis man prøver å gå ut av siden før man har trykket 'lagre', vil det dukke opp en melding om at endringene ikke har blitt lagret.



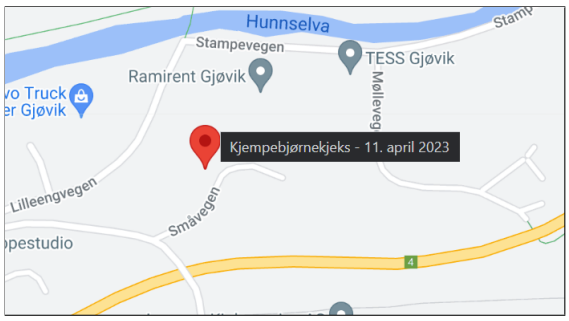
Figur 4.16: Advarsel om endringer som ikke er lagret

Ellers på siden er det et Google-kart som viser hvor planten befinner seg, og knapper som endrer status og offentlighet. Offentligheten til en henvendelse er bare relevant når den er markert som 'ferdig bekjempet', hvor posisjonen blir vist på et offentlig kart. Dette kartet har en egen side under 'offentlig kart' i venstremenyen på siden, som vist på figur 4.17.



Figur 4.17: Menyen hvor man kan navigere mellom nettsidene, inkludert begge versjonene av brukerskjemaet

Det offentlige kartet er kodemessig veldig enkelt, da det bare er et Google-kart med markører som viser type plante og dato, som vist på figur 4.18, og har som funksjon å vise at man lett kan publisere de henvendelsene man ønsker ut til offentligheten. Dette var et ønske av oppdragsgiver mot slutten av bacheloroppgaven, da de lurte på om det var mulig å finne en løsning på hvordan man kan vise fram bekjempede planter til andre utenfor innloggingssystemet. Et alternativ var og bare offentliggjøre det allerede eksisterende kartet, men dersom noen hadde sendt inn planter fra hagen sin eller andre private områder, så la vi til muligheten for å sette henvendelser private og offentlige, slik at bare de plantene som ønskes å vises av kommunen, blir vist offentlig.



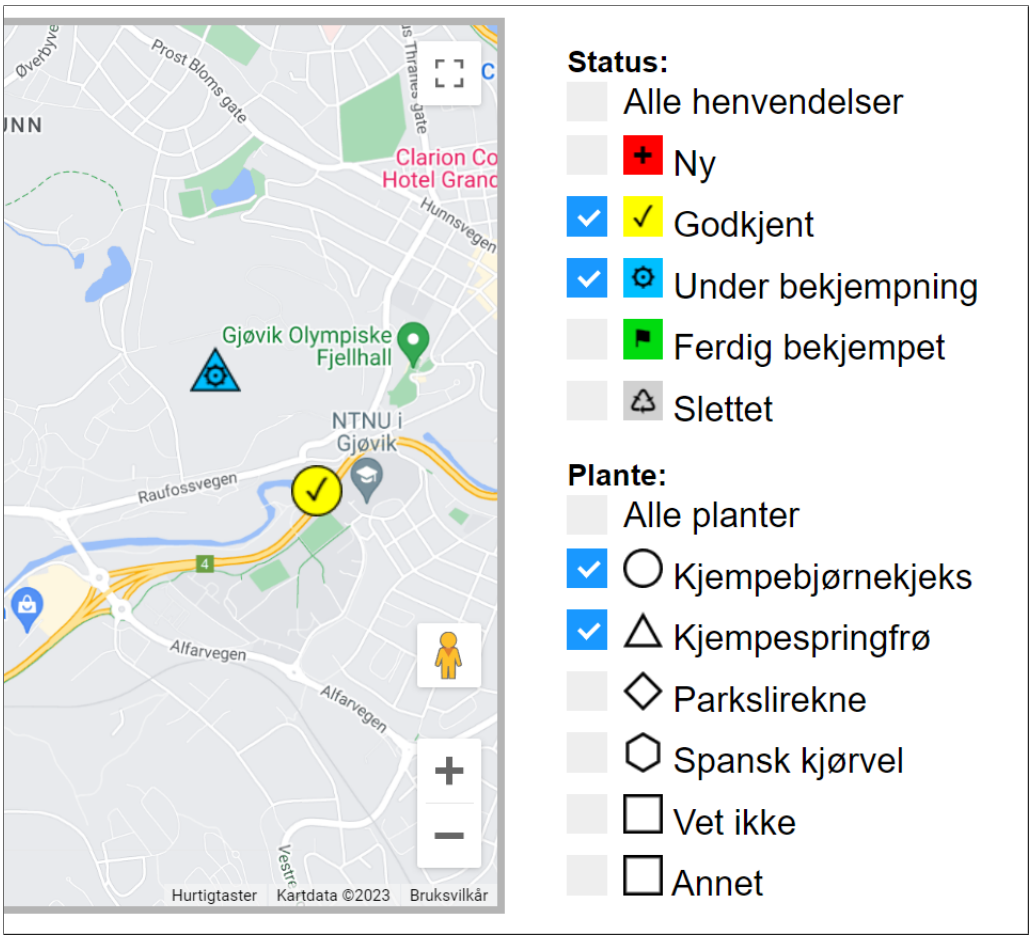
Figur 4.18: Et eksempel på en markør av en offentlig henvendelse på det offentlige kartet

Det andre kartet er 'kart' nettsiden, som bare er tilgjengelig for de som har tilgang til administrasjonssidene. I starten var denne siden veldig enkel og i prinsippet helt likt som 'offentlig kart', hvor den viste alle henvendelsene på et kart, men med forskjellige farger basert på hvilken status henvendelsen var.

Senere hadde vi et møte med oppdragsgiver og to til fra kommunen som jobbet med svartelistede arter. Vi ble enige om at først og fremst vil det være behov for filtrering av markører, da det etter en stund vil bli mange markører på en gang. I tillegg kom vi fram til at hvis vi brukte mer enn bare farger på kartet, vil det gjøre det mer tilgjengelig for fargeblinde, i tillegg til kartet vil få en bedre forståelse av hvilke planter som er hvor. Som vist på figur 4.19, så har planter hver sin geometriske figur som skiller dem fra hverandre, mens fargene skiller hvilken status de er. Det er i tillegg egne figurer inne i hver markør som samsvarer med fargene for å øke brukervennligheten til de som er fargesvake og fargeblinde.

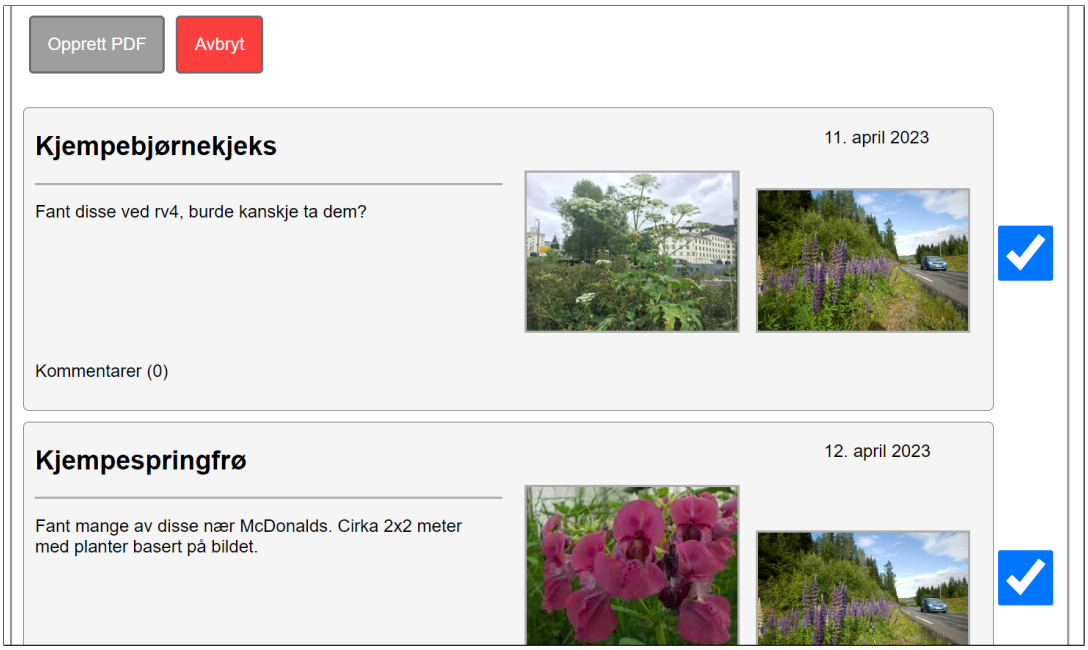
Figur 4.19: Egne markører for 'kart' nettsiden

Vi la også til muligheten å filtrere ut planter og statuser fra kartet, hvor man kan velge hvilke planter man vil vise samtidig som man velger hvilken status man vil vise. Som vist i figur 4.20, kan man for eksempel filtrere på godkjent og under bekjemping, og kjempebjørnekjeks og kjempespringfrø.



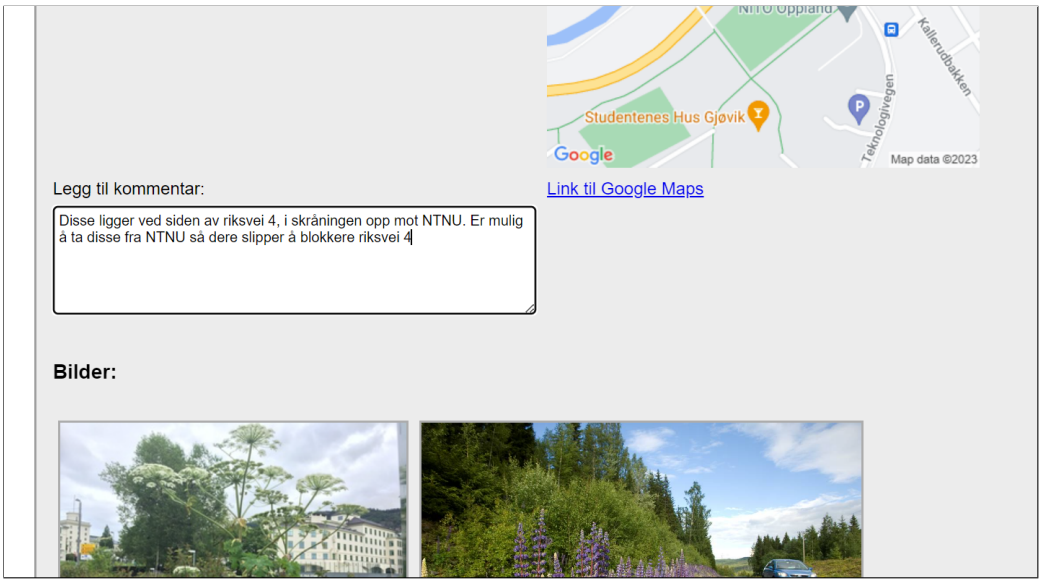
Figur 4.20: Eksempel på filtrering av kartmarkører

Når plantene skal bekjempes, må det være mulig å dele henvendelsene med de som skal ut å bekjempe plantene. Henvendelsene blir konvertert til en PDF, hvor det er viktig at all informasjon om henvendelsen er med, som type plante, beskrivelse og posisjon. I tillegg ville oppdragsgiver ha muligheten til å legge ved en kommentar skulle det være nødvendig på PDFen. Så før henvendelsen blir konvertet til en PDF, vil man ha muligheten til å skrive en kommentar som følger med PDFen. I figur 4.21 og 4.22, ser man at man kan velge hvilke henvendelser man ønsker å lage PDF fra, for så å legge til en kommentar på hver henvendelse hvis ønskelig. Det er mulig å opprette flere henvendelser på én PDF, som i figur 4.21 viser, men man kan også klikke inn på en henvendelse og opprette en PDF med én henvendelse.



Figur 4.21: Mulig å velge flere henvendelser samtidig for å opprette en PDF

Vi har også lagt inn cookie-funksjonalitet, slik at navigeringen mellom administrasjonssidene blir mer naturlig. Slik at hvis man er i 'godkjent' fanen og man trykker inn på en henvendelse, så er man fortsatt på 'godkjent' fanen når man går tilbake. I tillegg er det en cookie som er tilknyttet sortering og filtrering, igjen slik at når man går ut og inn igjen av henvendelsene så forblir sorteringen og filtreringen slik den var satt før man gikk ut av siden. Eksempelet i figur 4.23 viser til funksjonen som oppretter en ny cookie, som blir kalt hver gang man endrer fane eller sortering/filtrering.



Figur 4.22: Man kan legge til en kommentar på hver henvendelse før PDFen opprettes

```
// ====== COOKIE ====== //
//Set tab cookie
function setCookie(cookiename, value) {
  document.cookie = cookiename + "=" + value + ";"
}
```

Figur 4.23: En funksjon i Javascript som oppretter en ny Cookie

Kapittel 5

Implementering

5.1 Backend

Vi har allerede skrevet om designet av nettsidene, men bak det visuelle så ligger det en del kode. Vi vil i dette kapitlet presentere hvordan vi har valgt å implementere koden for å komme fram til en løsning slik at nettsidene fungerer i praksis.

5.1.1 Application Programming Interface

For nettsidene, har vi bygget en rekke APIer som blir brukt for overføring av data til og fra brukerskjemaet og administrasjonssidene. APIene blir nådd over HTTPS, som krypterer all data mellom bruker og server. I figur 5.1 er APIene til selve nettsidene, men det er også en del ekstra APIer som hjelper til med sending og mottakelse av data, som vist i figur 5.2. Et eksempel er når en person sender inn en henvendelse fra brukerskjemaet, så blir '/sendplant' kalt fra brukeren for å sende inn dataene til serveren som sender den videre til databasen.

```
//Website handlers
http.HandleFunc("/userform", frontend.UserformHandler)
http.HandleFunc("/newuserform", frontend.NewUserformHandler)
http.HandleFunc("/dashboard", frontend.AdminDashboardHandler)
http.HandleFunc("/henvendelser/", frontend.AdminInquiryHandler)
http.HandleFunc("/kart/", frontend.AdminMapHandler)
http.HandleFunc("/offentligkart", frontend.AdminPublicMapHandler)
http.HandleFunc("/login", frontend.AdminLoginHandler)
http.HandleFunc("/signup", frontend.AdminSignupHandler)
```

Figur 5.1: APIer for nettsidene

```
//Manage inquiry
http.HandleFunc("/runlogin/", admin.LoginHandler)
http.HandleFunc("/sendplant", userform.RecieveInquiry)
http.HandleFunc("/editinquiry", admin.EditInquiry)
http.HandleFunc("/addcomment", admin.AddComment)
```

Figur 5.2: APIer for håndtering av data

I tillegg til APIene for selve sidene, har vi egne APIer for statiske filer som CSS filene, Javascript filene og alle bildene som blir brukt på nettsidene (figur 5.3). Ettersom hver eneste statiske fil må hentes via 'http.FileServer(...)', kan vi bruke 'http.StripPrefix(...)' til å fjerne den virtuelle filbanen og legge til den faktiske filbanen til filen som skal hentes. Dette gjør det noe lettere å hente filer i koden, da man kan lage en helt egen virtuell filbane som er kort og enkel å huske på, som for eksempel 'static/images/' istedenfor den faktiske filbanen som kan inneholde mange mapper eller til og med endre seg underveis i utviklingen av prosjektet. Hvis filbanen til de statiske filene endrer seg, trenger man bare å endre det ett sted i koden, uansett hvor mange statiske filer man har linket til andre steder i koden.

```
//Admin css and js handlers
http.Handle("/static/adminjs/", http.StripPrefix("/static/adminjs/", http.FileServer(http.Dir("./frontend/admin/javascript"))))
http.Handle("/static/admincss/", http.StripPrefix("/static/admincss/", http.FileServer(http.Dir("./frontend/admin/css"))))
//Userform css and js handlers
http.Handle("/static/userformjs/", http.StripPrefix("/static/userformjs/", http.FileServer(http.Dir("./frontend/userform/javascript"))))
http.Handle("/static/userformcss/", http.StripPrefix("/static/userformcss/", http.FileServer(http.Dir("./frontend/userform/css"))))

//Image handlers
http.Handle("/static/images/", http.StripPrefix("/static/images/", http.FileServer(http.Dir("./frontend/admin/images"))))
http.Handle("/static/mapicons/", http.StripPrefix("/static/mapicons/", http.FileServer(http.Dir("./frontend/admin/mapicons"))))
```

Figur 5.3: APIer for statiske filer

5.1.2 Brukerskjema

Brukerskjemaet er ment for å være tilgjengelig for offentligheten. Selv om det er viktig å lage et godt design som er tilgjengelig og brukbart for alle potensielle brukere, er det også viktig å tenke på skadelige handlinger som kan bli gjort på systemet. Ettersom hvem som helst skal ha tilgang, er det viktig å sette noen begrensinger for å hindre at noen prøver å ødelegge serveren.

Ett eksempel kan være en bruker som vil prøve å være litt morsom, ved å laste opp bilder som er flere gigabytes. Først og fremst er det lagt inn sperrer i via frontend, som forklart i punkt 5.2, men en som er litt mer teknisk vet at man lett kan komme seg rundt sperrer som er lagt inn via Javascript og HTML. Derfor er det også viktig å legge inn visse sperrer på selve serveren også. I figur 5.4, kan man se at av de første tingene som blir sjekket når en henvendelse mottas av serveren, er om størrelsen er under den tillate maksstørrelsen som er satt.

```
//Set a 35 MB limit to post request
sizeLimit := int64(35 * consts.MB)
_, err := ioutil.ReadAll(http.MaxBytesReader(nil, readBody, sizeLimit))
if err != nil {
    println("Request body is too large for size limit: ", sizeLimit)
    return
}
```

Figur 5.4: Sjekker størrelse på henvendelsen før den blir tatt med videre

Dersom henvendelsen er gyldig og blir sendt igjennom, vil serveren legge til en del informasjon, før den legges inn i databasen. Den setter blant annet et datostempel på henvendelsen, angir en unik ID og dekoder bilder som har blitt enkodet til base64 [18] for overføring over nettet.

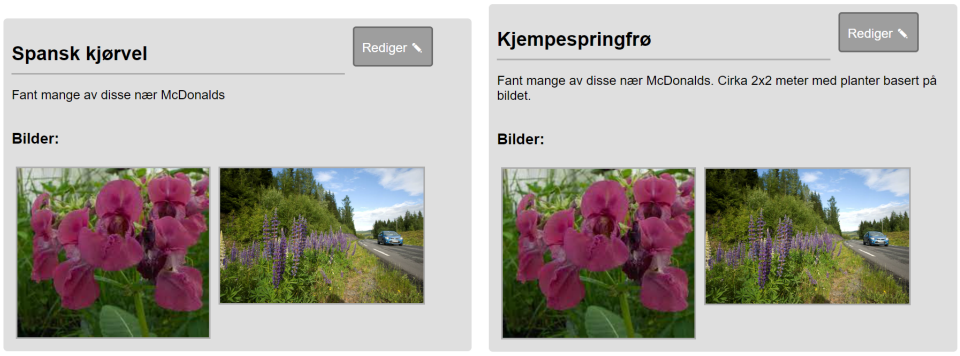
5.1.3 Admin

For administrasjonssidene, hadde vi tett kommunikasjon med oppdragsgiver for å sørge for at funksjonene ble implementert slik de forestilte seg. Selve designet på de diverse nettsidene og funksjonene, har endret seg underveis, som beskrevet i punkt 4.1 om design, men det har også implementeringen av diverse funksjoner.

Alle nettsidene som tilhører administrasjonssidene, skal være utilgjengelig for offentligheten. Vi har utbredet et brukersystem hvor man må logge inn, men hvis dette skulle faktisk blitt implementert inn i Gjøvik kommune sine nettsider, så ville vi heller ha integrert kommunen sitt eget brukersystem istedenfor å bruke vårt eget. Dermed ble vår egenlagde innlogging nedprioritert, og vi brukte istedenfor innloggingen vår til å vise at en innlogging kunne fint blitt integrert med resten av prosjektet.

Administrasjonssidene har to hovedsider som det meste av funksjonalitet. Den første er ’henvendelser’, som inneholder alle henvendelsene fra brukere som har sendt inn angående svartelistede planter. Det er her alle henvendelsene fra brukere vil være, og her de blir administrert og sett over. Backenddelen består hovedsakelig av en rekke APIer som blir kalt for å endre status på en henvendelse, f.eks. godkjenne en ny henvendelse eller sette en henvendelse under bekjempning til ferdig bekjempet. I tillegg er det APIer for endring av en henvendelse, dersom noen har sendt inn feil navn på planten eller skrevet noe som må rettes opp, så kan man gå inn og endre dette i etterkant, som vist i figur 5.5.

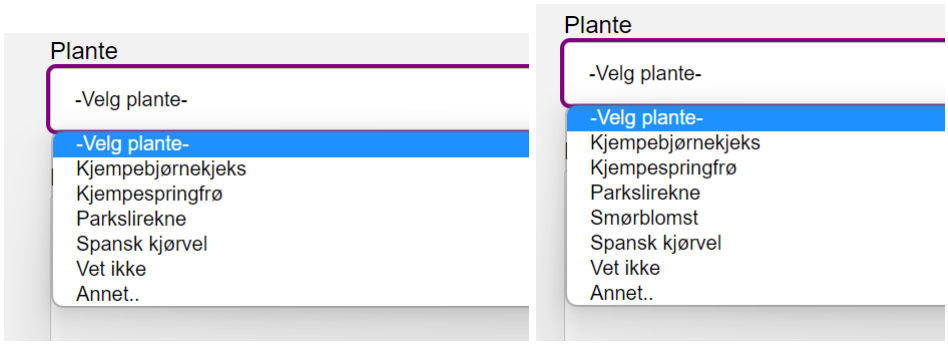
’Kart’ nettsiden er også sentral i oppgaven, men det eneste den henter fra backend er alle henvendelsene, resten gjøres i Javascript. Mer om dette er skrevet i punkt 5.2 om frontend.



Figur 5.5: Henvendelse: Før og etter redigering av navn og beskrivelse

Det er også en side til, 'dashboard', som ikke har mye implementert. Hovedfunksjonen til denne siden var å vise at plantene som er lagt inn i systemet ikke trenger å være fast. Selv om oppgaven er for Gjøvik kommune, vil vi vise at det er fullt mulig å tilpasse den for andre kommuner som har andre planter de vil bekjempe. Som vist i figur 5.6 og 5.7, så er det veldig enkelt å legge til og fjerne planter. Dette var interessant for oppdragsgiver, men det var ikke et sterkt behov for dette, da det er sjeldent at nye svartelistede planter plutselig oppstår som krever at man legger til nye planter. Dermed ble implementeringen veldig enkel, for å vise konseptet uten å bruke for mye tid på noe som ikke var sterkt nødvendig.

Figur 5.6: Dashboard: Legge til en ny plante, smørblomst



Figur 5.7: Brukerskjema: Før og etter 'smørblomst' er lagt inn

5.2 Frontend

5.2.1 Brukerskjema

Som nevnt tidligere i punkt 4.1 om design, har vi implementert to forskjellige design på brukerskjemaet vårt. Selv om de er forskjellige, er de i prinsippet helt likt implementert. Den største forskjellen er at det flersidige brukerskjemaet har noen ekstra linjer med Javascript kode som håndterer hvilken side som skal vises til enhver tid.

For å prøve å redusere antall steg en bruker må gå igjennom for å sende inn et skjema, har vi lagt til muligheten for å lese av posisjonsdata fra bilder. Når brukeren laster opp et bilde, vil posisjonsdataene leses fra det første bildet som blir lastet opp. Selv om ikke alle har dette aktivert på telefonen sin, så er geotagging som oftest aktivert på telefoner som standard [19]. Dermed er det en fin balanse mellom de som er teknologiske nok til å skru av geotagging, vil ikke ha noe problem med å tillate nettleseren på mobilen til å dele posisjonen sin. De som ikke er like teknologiske av seg, får litt mer assistanse da de slipper å dele posisjon ettersom den er funnet automatisk fra bilder. Skulle posisjonen bli lest ut ifra et bilde eller ikke, gis det beskjed om posisjonen trengs eller om den burde dobbeltsjekkes etter å ha bli hentet fra et bilde.

Det er også lagt inn en del sperrer på hva man kan sende inn, på lik linje med backend som skrevet i punkt 5.1. Ettersom alle disse sperrene ligger i HTML og Javascript, kan de lett bli ignorert ved å skrive om litt i den lokale koden på nettsiden, ved bruk av utviklerverktøy fra nettleseren [20]. Hovedoppgaven til disse sperrene er ikke å stoppe noen i å gjøre hærverk, det er backend sin jobb, men heller å hjelpe brukerne i og ikke gjøre noe som trigger sperrene i backend. For eksempel, hindre brukere fra å laste opp for store bilder, som vil resultere i at henvendelsen aldri blir sendt, men istedenfor gir dem en beskjed om at henvendelsen ikke kan sendes fordi bildene er for store.

5.2.2 Admin

For administrasjonsdelen har Javascript en stor del i funksjonaliteten på sidene. Nettsidene 'kart' og 'offentlig kart' håndteres hovedsakelig av Javascript. Først hentes alle henvendelsene ut i en for-loop ved hjelp av 'HTML/template' pakken, som vist i figur 5.8. Her blir informasjonen hentet ut, men med style='display:none'", da vi ikke er interessert i informasjonen i rent tekstformat. Først skal vi prosessere dataene vi har hentet inn i Javascript, som vil dermed legge til koordinater på kartet. Denne metoden med å hente informasjon ved bruk av 'HTML/template' pakken, brukes også mye på 'henvendelser' nettsiden. I tillegg til for-loppen, bruker vi if-setninger for å bestemme hva som vises basert på hva henvendelsen. For eksempel, basert på om henvendelsen er ny/godkjent/etc..., eller har vedlagt bilder eller ikke, som vist i figur 5.9.

```
<!--Adds coordinates for displaying markers-->
{{range $elem := .List}}
<p class="coord" style="display:none;"
onclick="placeMarker('{{$elem.Status}}', '{{$elem.Lat}}', '{{$elem.Lng}}',
'{{$elem.Listname}}', '{{$elem.StringDate}}', '{{$elem.Id}}')"></p>
{{end}}
```

Figur 5.8: En for-loop som går igjennom alle henvendelser med 'HTML/template' pakken fra Golang

```
{{if not $elem.ImgNames}}
<h3 class="maxWidth">Ingen bilder</h3>
{{else}}
<h3 class="maxWidth">Bilder:</h3>
{{end}}
<div class="displayImg maxWidth">
  {{range $imgelem := .ImgNames}}
  <img src="{{$imgelem}}" class="showImg">
  {{end}}
  {{end}}
</div>
```

Figur 5.9: En if-setning som enten legger til teksten "Ingen bildereller Bilder:bb-asert på om det er vedlagt bilder med henvendelsen eller ikke

All informasjon fra 'HTML/template' pakken hentes inn ved innlastning av nettsiden, så dersom noe fra databasen oppdaterer seg imens man har en nettside oppe, så vil ikke den nye informasjonen vises på siden før den oppdateres. For å hindre unødvendig mange oppdateringer, så bruker vi Javascript for å gjøre endringene lokalt slik at vi slipper å kjøre en oppdatering. Dette hjelper med belastning på server, i tillegg til unødvendig tid brukt på oppdatering av siden, spesielt hvis man har tregt internett og man må laste inn nye bilder og data fra databasen hver gang man gjør en endring. Dette brukes for eksempel når man skal gjøre en endring på en henvendelse. Eksempelet i figur 5.10, viser til at vi bruker Javascript til å endre nettsiden, samtidig som vi sender inn data til API om endring av dataene.

```
//Edit description
document.getElementById("plantDesc").style.display = "block";
document.getElementById("plantDescInp").style.display = "none";
document.getElementById("plantDesc").innerHTML = document.getElementById("plantDescInp").value;

// -- Send POST request to change inquiry data --        You, now • Uncommitted changes
var xhr = new XMLHttpRequest();
var formData = new FormData(form);
//Open the request
var url = window.location.href.split("/");
xhr.open("POST",url[0]+"//"+url[1]+url[2]+"/editinquiry");
xhr.setRequestHeader("Content-Type", "application/json");
```

Figur 5.10: Dataene som ble endret, endres først i Javascript, dermed sendes dataene inn til APIen på serveren

5.3 Database

Vi har en egen IaaS løsning i SkyHigh, i.e., Openstack på NTNU, for databasen. Av sikkerhetsmessige årsaker valgte vi å ha databasen på en separat løsning fra der applikasjonen er implementert. På denne måten hadde vi mer kontroll og hvis noe skulle skje med en av løsningene ville ikke den andre bli påvirket. Vi bruker Open Source relasjonsdatabasen MariaDB Server, og nedfor vil vi forklare hvordan denne databasen er implementert i prosjektet vårt.

5.3.1 MariaDB på Ubuntu

Vi har installert MariaDB på Ubuntu med pakken 'mariadb-server'. Merk at på tidspunktet for installasjonen var MariaDB version 10.3 tilgjengelig fra Ubuntu 20.04 standard APT repositories. Vi var opptatt av å konfigurere MariaDB litt sikrere enn standard innstillingene som var satt. Til å begynne med, kjørte vi sikkerhets-scriptet 'mysql_secure_installation' som var inkludert i pakken. Dette scriptet fjernet en del usikre standardinnstillinger, e.g., ekstern root pålogging og eksempelbrukere. Vi så at standard root-brukeren i MariaDB autentiserer seg med 'unix_socet plugin', dette kunne by vanskeligheter hvis programmet vårt som er eksternt trengte administrative rettigheter når den kommuniserte med databasen. Derfor laget vi en administratorbruker med samme rettigheter som en root-bruker bare at den autentiseres med passord. Per i dag bruker programmet strengt tatt ingen root-rettigheter unntatt CRUD, og vi kunne derfor gjort det sikrere ved å la programmet bruke en bruker som *bare* hadde disse rettighetene. Vi begrenset derimot administratorbrukeren til at den bare fikk kommunisere med den spesifikke databasen vi lagde for programmet vårt, altså ingen andre databaser vi eventuelt hadde på serveren.

Databasen vi satte opp for programmet ble kalt 'svartelistedb'. Databasen har sju tabeller, og er strukturert slik domenemodellen vår viser i punkt 2.2 om kravspesifikasjon. Figur 5.11 viser hvordan enheten 'PlantInquiry' i domenemodellen ser ut i praksis, i.e., hvordan den er satt opp på databasen. Blant annet viser figuren at ingen av verdiene til en ny rad i 'PlantInquiry' kan være NULL, da en henvendelse må ha alle disse verdiene når den lagres på databasen. Alle verdiene som er av type varchar har et begrenset antall karakterer også, dette hjelper på å spare plass i databasen. 'id' er brukt som primærnøkkel (PRI) i denne tabellen, som unikt definerer alle henvendelsene.

```
MariaDB [svartelistedb]> SHOW COLUMNS FROM PlantInquiry;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| id          | varchar(20)  | NO   | PRI | NULL    |       |
| listname    | varchar(80)  | NO   |     | NULL    |       |
| name        | varchar(80)  | NO   |     | NULL    |       |
| description | varchar(300) | NO   |     | NULL    |       |
| mail        | varchar(80)  | NO   |     | NULL    |       |
| lat         | varchar(80)  | NO   |     | NULL    |       |
| lng         | varchar(80)  | NO   |     | NULL    |       |
| status      | varchar(80)  | NO   |     | NULL    |       |
| date        | date         | NO   |     | NULL    |       |
| stringDate  | varchar(80)  | NO   |     | NULL    |       |
| private     | tinyint(1)   | NO   |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
11 rows in set (0.003 sec)
```

Figur 5.11: Tabell 'PlantInquiry' slik den er satt opp på databasen 'svartelisteDB'

Figur 5.12 viser tabellen 'PlantComment' slik den er satt opp i databasen. Det som er spesielt med denne tabellen er at den er en svak enhetstype som er avhengig av 'PlantInquiry' enheten. Dette er har vi spesifisert ved å lage en fremmednøkkel som referer til 'id' fra 'PlantInquiry'. Disse tabellene er også linket slik at ved sletting eller oppdatering av foreldretabellen, i dette tilfellet 'PlantInquiry', vil påvirke dattertabellen 'PlantComment'. Vi har implementert samme løsning for bilder og thumbnails relatert til en henvendelse.

```
MariaDB [svartelistedb]> SHOW COLUMNS FROM PlantComment;
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| id         | varchar(20)  | NO   | MUL | NULL    |       |
| comment    | varchar(300) | NO   |     | NULL    |       |
| date       | date         | NO   |     | NULL    |       |
| stringDate | varchar(80)  | NO   |     | NULL    |       |
| userId     | varchar(30)  | NO   |     | NULL    |       |
+------------+--------------+------+-----+---------+-------+
5 rows in set (0.003 sec)
```

Figur 5.12: Tabell 'PlantComment' slik den er satt opp på databasen 'svartelisteDB'

5.3.2 MariaDB med Golang

Vi kommuniserer med MariaDB-databasen ved å bruke Golang-pakken 'database/sql'. Denne pakken trenger også en driver for at vi kan få bruke alle funksjonalitetene dens, men mer om dette kan bli lest i punkt 6.5 om teknologier. Figur 5.13 viser starten på programmet vårt, hvor vi setter opp en database handle som vi har satt til å være en global variabel. Vi har sensurert en del av 'sql.Open()'-parameterne da dette er påloggingsinformasjonen til brukeren vi har lagd for databasen. Hvis adgangen til databasen feiler må programmet stoppe helt, og derfor bruker vi 'log.fatal' som tvinger programmet til å gå ut. Figuren viser også at vi etter å ha lagd en database handle sjekker vi at databasen er opp med funksjonen 'database.DatabaseStatus()'. Status sjekkes enkelt ved at vi loggfører versjonen av MariaDB som databasen bruker. Etter dette er programmet klar for neste steg som er å sette opp selve nettsiden.

```go
// DATABASE
// Create the database handle, confirm driver is present
database.DB, err = sql.Open("mysql", "[redacted]@tcp(10.212.170.74:3306)/svartelistedb?parseTime=true")
if err != nil {
    log.Fatal(err)
}
defer database.DB.Close()
// Check status for database
database.DatabaseStatus()
```

Figur 5.13: Go-kode som lager en database handle for å kommunisere med MariaDB serveren vår

Heretter vil vi se nærmere på implementeringen av ulike CRUD-oppgaver som programmet utfører på databasen. Figur 5.14 viser Golang-koden hvor vi henter en henvendelse fra databasen med en bestemt 'id'. Vi starter med å spørre etter henvendelsen i 'PlantInquiry' tabellen, ved å bruke 'QueryRow()'-funksjonen fra 'database/sql'-pakken. Pakken tilbyr også en måte å separere SQL-setningen fra verdien 'id', og dermed stoppes det for SQL-injeksjon. Dette gjøres ved å bruke et '?' i stedet for å legge til 'id string' direkte inn i SQL-setningen. Det at vi spesifi-

serer kolonnene vi vil ha istedenfor å bruke tegnet '*' som henter det samme er også en sikkerhetspraksis som vi følger. På dette viset har vi mer kontroll over hva som hentes fra databasen og dermed blir programmet sikrere. Videre putter vi resultatet fra spørring i en struct vi har definert som 'inq'. Eventuelle feilmeldinger under SQL-spørringen blir loggført på innsiden med 'log.Println()', i.e., brukeren av nettsiden skal ikke få se interne feilmeldinger som dette. Nå er henvendelsen fra 'PlantInquiry' tabellen hentet, men vi skal også hente alle eventuelle bilder, thumbnails, og kommenterer som er knyttet til denne henvendelsen. Dette gjøres ved å kalle på de tre funksjonene 'ListPlantImages', 'ListPlantThumbnails', og 'ListPlantComments'. Implementeringen av sistnevnte funksjon er vist i figur 5.15, og vi skal se nærme på den under. Den initialiserte structen blir til slutt sendt tilbake.

```go
// Return a single plant inquiry
func GetPlant(id string) *structs.PlantInquiry {
    var inq structs.PlantInquiry

    if err := DB.QueryRow(`SELECT id, listname, name, description, mail,
        lat, lng, status, date, stringDate, private
        FROM PlantInquiry
        WHERE id = ?`,
        id).Scan(&inq.Id, &inq.Listname, &inq.Name, &inq.Desc, &inq.Mail,
        &inq.Lat, &inq.Lng, &inq.Status, &inq.Date, &inq.StringDate, &inq.Private); err != nil {
        log.Println(fmt.Errorf("GetPlant: %v", err))
    }

    inq = ListPlantImages(inq)     // Retrieve images
    inq = ListPlantThumbnails(inq) // Retrieve thumbnails
    inq = ListPlantComments(inq)   // Retrieve comments

    return &inq
}
```

Figur 5.14: Go-kode som henter én henvendelse fra databasen

Implementeringen i figur 5.15 er relativt lik SQL-spørringen i figur 5.14 hvor vi henter én henvendelse. I dette tilfellet skal vi hente alle radene i tabellen 'PlantComment' som er knyttet til en bestemt henvendelse. Vi bruker funksjonen 'Query()' til 'database/sql'-pakken ved spørringer som henter ut flere rader. I WHERE-klausulen her er 'id' hentet fra den tilsendte structen, men dette er da den samme 'id'-en som er tilsendt i 'GetPlant'-funksjonen i figur 5.14. Vi henter alle radene først, deretter looper vi gjennom hver rad og putter resultatet i structen, som sendes tilbake ved funksjonsslutt. Merk at her kan det dukke opp feilmeldinger på flere steder, og disse blir loggført på innsiden.

```go
// Return all plant comments for an inquiry
func ListPlantComments(inq structs.PlantInquiry) structs.PlantInquiry {
    rows, err := DB.Query(`SELECT
    id, comment, date, stringDate, userId
    FROM PlantComment WHERE id = ?`, inq.Id)
    if err != nil {
        log.Println(fmt.Errorf("ListPlantComments: %v", err))
    }
    defer rows.Close()

    // Loop through rows
    for rows.Next() {
        var com structs.Comment
        if err := rows.Scan(&com.Id, &com.Comment, &com.Date,
            &com.StringDate, &com.UserId); err != nil {
            log.Println(fmt.Errorf("ListPlantComments: %v", err))
        }
        inq.Comments = append(inq.Comments, com)
    }
    if err = rows.Err(); err != nil {
        log.Println(fmt.Errorf("ListPlantComments: %v", err))
    }

    return inq
}
```

Figur 5.15: Go-kode som henter alle kommentarene knyttet til en henvendelse fra databasen

Når en ny henvendelse skal legges til databasen kaller vi på funksjonen 'InsertPlant', og implementeringen til denne er vist i figur 5.16. Ved innsetting av nye rader bruker vi funksjonen 'Exec()' av 'database/sql'-pakken. Her er vi også strenge på at vi ikke legger verdiene direkte i SQL-setningen, men linker de opp til '?' slik at vi minker risikoen for SQL-injeksjon. Vi må gjøre lignende operasjon for bildene, thumbnails, og eventuelle kommentarer knyttet til den nye henvendelsen. Dette gjøres i de tre funksjonene 'InsertPlantsImages', 'InsertPlantThumbnails', og 'AddComment'. Sistnevnte funksjon skal ikke kjøres ved mindre henvendelsen har kommentarer, og loopen tar for seg hver kommentar og legger de til databasen.

```go
// Insert new plant inquiry into DB
func InsertPlant(plant structs.PlantInquiry) {
    _, err := DB.Exec(`INSERT INTO
        PlantInquiry (id, listname, name, description, mail,
        lat, lng, status, date, stringDate, private)
        VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)`,
        plant.Id, plant.Listname, plant.Name, plant.Desc, plant.Mail,
        plant.Lat, plant.Lng, plant.Status, plant.Date, plant.StringDate, plant.Private)
    if err != nil {
        log.Println(fmt.Errorf("InsertPlant: %v", err))
    }
    // Insert images
    InsertPlantImages(plant.Id, plant.ImgNames)
    // Insert thumbnials
    InsertPlantThumbnails(plant.Id, plant.Thumbnails)
    // Insert possible comments
    if len(plant.Comments) != 0 {
        for i := range plant.Comments {
            AddComment(plant.Comments[i])
        }
    }
}
```

Figur 5.16: Go-kode som lagrer én ny henvendelse i databasen

Til slutt vil vi se på et eksempel hvor vi oppdaterer én henvendelse på databasen. Figur 5.17 viser implementeringen av hvordan vi oppdaterer statusen til en bestemt henvendelse til å bli 'Processing', i.e., 'Under behandling'. Her bruker vi også 'Exec()'-funksjonen til 'database/sql'-pakken, og er presis i SQL-språket vårt. Legg merke til at også her er eventuelle feilmeldinger loggført med funksjonsnavn og feilmeldingen som dukket opp. Hvordan vi håndterer feilmeldinger er likt for alle funksjoner som innebærer bruk av databasen, slik at lett kunne se hvor feilen oppstod under kjøring. Lignende løsninger som den i figur 5.17 er implementert for å oppdatere andre egenskaper til en henvendelse lagret på databasen.

```go
// Update DB with processing plant
func ProcessingPlant(id string) {
    _, err := DB.Exec("UPDATE PlantInquiry SET status = 'processing' WHERE id = ?", id)
    if err != nil {
        log.Println(fmt.Errorf("ProcessingPlant: %v", err))
    }
}
```

Figur 5.17: Go-kode som oppdaterer én henvendelse i databasen med status 'processing'

5.3.3 Synkronisering

Vi var opptatt av at brukerskjemasiden og administrasjonssidene var synkron, slik at henvendelser som ble sendt inn dukket opp med én gang på administrasjonssidene for behandling. I tillegg ville vi at løsningen skulle kunne kjøres på flere maskiner samtidig, e.g., to administrasjonsbrukere er logget inn og aktiv på hver sin front. Vi har derfor implementert løsning slik at innholdet som brukeren jobber med under programkjøringen er live med det nyeste datainnholdet fra databasen. Endringer gjort på en henvendelse for eksempel blir alltid lagret fortløpende på databasen, og vil derfor være tilgjengelig for andre aktive brukere med en gang. Dette er derimot ikke lønnsomt ved tanke på antall forespørsler vi gjør til databasen i løpet av programkjøringen. For å vise til et eksempel, så henter vi alle henvendelsene hver gang en bruker trykker knappen 'Henvendelser' for å liste henvendelsene gjennom administrasjonssidene. Akkurat denne operasjonen kan risikere å bli farlig sakte hvis det er mange henvendelser som skal hentes fra databasen. Mer om denne diskusjonen skriver vi om i punkt 9.1.2 i avslutningen.

Kapittel 6

Teknologier

6.1 Golang

For å kunne bygge nettsidene, har vi hatt bruk for en rekke teknologier. I dette kapitlet vil vi gå over hvilke teknologier vi har valgt og se på hvorfor vi valgte dem.
Vi har valgt å bruke programmeringsspråket Golang [21] for serveren vår, som kommer med flere funksjonaliteter som er nyttig for oppgaven. Golang har en innebygd pakke for bruk av HTML, mer om det i punkt 6.2, som vil tillate oss å bygge frontend inn i serveren, i tillegg til at det er et språk som gjør det lettvint å lage APIer og egendefinerte datastrukturer for utsending av data.

Begge gruppemedlemmene har hatt emnet 'PROG2005 - Cloud Technologies' og 'PROG2006 - Avansert Programmering', hvor vi ble introdusert for Golang og jobbet mye med språket. Ettersom begge to har fått god erfaring med Golang, i tillegg til at vi har blitt glade i hvor fleksibelt og intuitivt det er, var det ingen tvil om at Golang var en god kandidat for denne oppgaven.

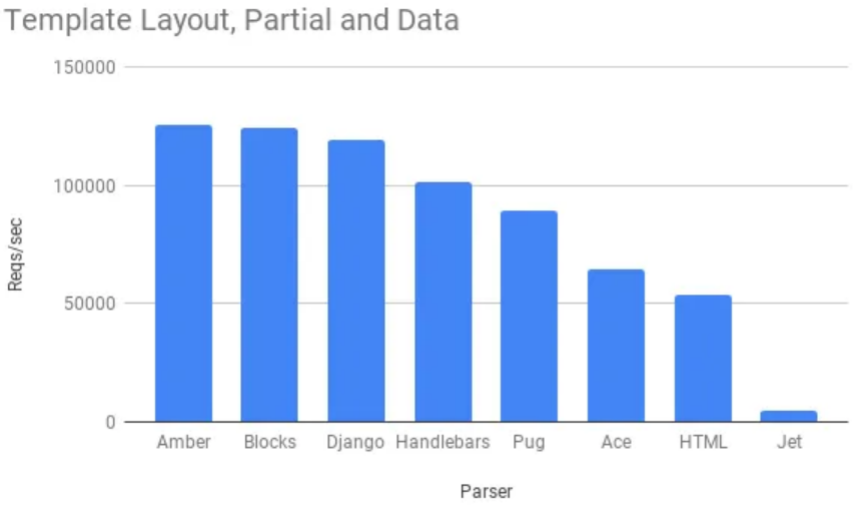
Golang har også god støtte for å kunne kjøre flere prosesser på en gang, som gjør det lettere å kunne bygge ut serveren hvis det er behov for håndtering av større trafikk. Ettersom oppgaven vår omhandler én kommune som bare mottar noen få henvendelser i uken, så vil ikke dette være relevant for oss under prosjekttiden, men det ville gjort det enklere å bygge ut for flere og større kommuner skulle vi ha bygget videre på prosjektet.

Selv om trafikken ikke er stor, så vil fart og effektivitet alltid være positivt. Golang er et av de raskeste og mest energi-effektive språkene [22][23] sammenlignet med andre høy-nivå programmeringsspråk som Python [24], C++ [25] og Java [26]. Ett av poengene med oppgaven er å samle all informasjon på ett sted, slik at kommunen kan slippe å bruke tid på unødvendig samling av informasjon fra ulike steder. Dersom Golang kan spare noen tidels sekunder for hver interaksjon med serveren, så blir det fort mange sekunder spart på slutten av dagen.

6.2 Golang HTML/Template

'HTML/template' pakken fra Golang [27], er en pakke som er innebygd i Golang språket. Den bygger frontend delen av nettsiden slik at man får en 'fysisk' nettside som brukere kan få tak i, ved hjelp av HTML, CSS og Javascript filer. Ettersom pakken er innebygd i Golang, tok begynnelsen av oppgaven betraktelig mindre tid da vi slapp problemer med installering av tredjepartsprogrammer og ekstra feilsøking når vi skulle begynne å koble backend med frontend.

Selv om hovedgrunnen til valget av 'HTML/template' var at det var enkelt å sette opp, så er det i tillegg en veldig godt egnet 'template engine' for vårt bruk. Til tross for at 'HTML/template' pakken ikke er den aller beste, når det kommer til fart eller effektivitet, så er den mer enn god nok for vår hensikt. Som vist på figur 6.1, hentet fra en test på Medium.com [28], så kan man se at 'HTML/template' pakken er en av de tregeste blant alternativene. Ettersom dette er for svartelistede planter, som mottar i dag omtrent 10-20 henvendelser i måneden, så har ikke Gjøvik kommune et behov for å tilfredsstille flere tusen forespørsler om gangen. Altså er tiden vi sparer på oppsett og installering verdt det, med tanke på det vi taper på dårligere ytelse.



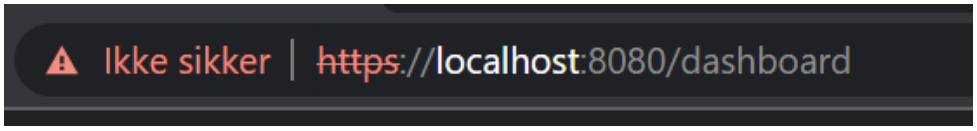
Figur 6.1: Ytelse til ulike 'template engines', hvorav 'HTML' er 'HTML/template' pakken vi brukte

6.3 Hosting

For å kjøre serveren utenom localhost, har vi valgt å implementere løsningen vår på en IaaS løsning i Openstack på NTNU, i.e., SkyHigh Gjøvik [29]. Vi har også brukt SkyHigh for databasen vår, se punkt 5.3 om Implementering for mer informasjon om dette. Gjøvik kommune hadde ingen spesifikke krav på hvor oppgaven ble implementert, så SkyHigh ble naturlig førstevalget vårt da det er et lett og nyttig verktøy som er gratis for NTNU studenter.

Selv om Skyhigh er en god løsning som gir nettsiden et mer realistisk preg når det gjelder å kjøre lokalt på maskinene våre, så er ikke Skyhigh helt ideelt. Det største problemet er at man må være koblet til NTNU sine nettverk for å få tilgang til nettsiden når den kjører på en Skyhigh server. Dette er ikke et problem for oss som enten er fysisk på campus eller bruker VPN, men for andre som ikke har tilknytning til NTNU, vil ikke ha tilgang. Alternativer til Skyhigh kunne vært Google Cloud web hosting eller Amazon Web Services, hvor Gjøvik kommune hadde hjulpet med et lite budsjett. For vår del var ikke dette nødvendig, da hensikten med oppgaven var å lage noe som kunne bli lagt inn på Gjøvik kommune sine nåværende nettsider. Altså var det viktigere for oss å kunne vise fram en nettside, enn å bruke tid på hvor nettsiden kjører.

En fordel med Skyhigh, er at nettsiden oppfører seg litt annerledes enn om man kjører den lokalt på maskinen. For eksempel, tillater ikke Google Chrome at man deler posisjonsdata med en nettside som bare bruker HTTP. De aller fleste nettsider i dag bruker HTTPS, inkludert Gjøvik kommune sine nettsider, som betyr at vi kan sette opp HTTPS på serveren når den kjører på Skyhigh. Da kan vi teste ut sikkerhetsmekanismer som er tatt i bruk på ulike nettlesere og enheter, slik at vi er sikre på at alle funksjoner er oppe og går i et realistisk scenario. Alt som må til er å lage ett eget SSL sertifikat for localhost [30], som gir oss en HTTPS kobling til serveren. En av ulempene er at man får en advarsel om utrygt nettsted, ettersom 'localhost' ikke er en unik nettadresse [31], som vist i figur 6.2. Dette er derimot ikke et problem, da den bare gir en advarsel som kan forbigås, og ellers oppfører seg som en normal HTTPS nettside som krypterer dataene til og fra.



Figur 6.2: Advarsel som gis når man kobler til localhost med HTTPS

6.4 Docker

Vi valgt å implementere løsningen vår ved bruk av Docker [32] på SkyHigh. Vi ble introdusert til Docker gjennom emnet 'PROG2005 - Cloud Technologies', og så på det som effektivt verktøy som kunne forenkle implementeringen av løsningen for fremtidige kommuner. Vi kunne ikke forutsi hvilke IT-systemer kommunene i Norge brukte, så vi ville ha et verktøy som kunne automatisere implementeringen og forenkle prosessen av å ta i bruk løsningen uansett hvilket system de hadde. Vi endte derfor opp med Docker. Det er et kjent verktøy som vi mente også kommuner kunne ta i bruk, og med en godt skrevet Dockerfil ville ikke kommunen trenge å gjøre annet enn å kjøre filen med Docker for å containerize applikasjonen og ta den i bruk. I tillegg kunne kommunen i teorien endre Dockerfilen slik at den passet bedre til deres system om nødvendig.

Vi oppdaget derimot flere fordeler med Docker under utviklingsprosessen. Vi laget en Dockerfil til prosjektet allerede i oppstartsfasen. På denne måten fikk vi forsikre oss om at applikasjonen var mulig å implementeres under hele utviklingen, samt at vi fikk effektivisert testprosessen. Hver gang vi nådde en mileperle eller hadde gjort store endring i prosjektet, laget vi og kjørte en ny Docker container på SkyHigh-serveren. Slik fikk vi forløpende testet applikasjonen etter endringene som var gjort fra forrige versjon av containeren. Mot slutten var vi på vår 15. versjon. Merk at ved å teste applikasjonen som var implementert på serveren, oppdaget vi også flere bugs som ikke hadde blitt sett lokalt.

6.5 Database

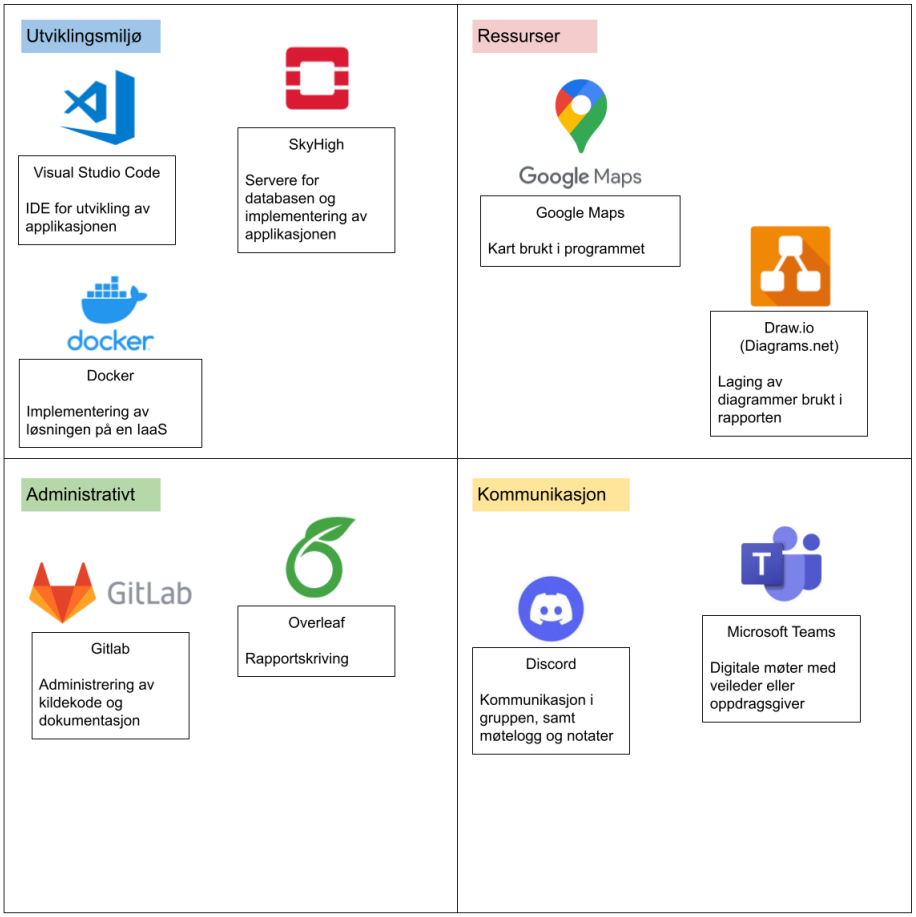
Valg av database var påvirket av innholdet vi skulle lagre og hvorvidt databasemiljøet var kompatibelt med Skyhigh og Golang. Vi hadde relativt lite innhold som måtte lagres og derfor lette vi etter en enklere løsning som dekket vårt 'lille' behov, i.e., vi trengte ikke en avansert databaseløsning med unødvendige funksjonaliteter. Vi var også opptatt av at databasen skulle være rask å sette opp siden vi har i tidligere prosjekter hatt problemer med installasjon av forskjellige Open Source alternativer, e.g., CockroachDB [33]. Vi foretrakk at installasjonen var mulig på en Ubuntu-løsning i SkyHigh, så vi kunne unnslippe og flette inn enda et tredjepartssystem med løsningen vår.

Vi har erfaring fra både relasjonsdatabaser og ikke-relasjonsdatabaser fra emnene 'IDATG2204 - Datamodellering og databasesystemer' og 'PROG2005 - Cloud Technologies', og i dette prosjektet syntes vi en relasjonsdatabase passet godt. I punkt 2.2 om kravspesifikasjon nevnte vi at en henvendelse er midtpunktet som knytter majoriteten av partene i prosjektet sammen, det var derfor naturlig at vi valgte en løsning hvor disse relasjonene kunne spesifiseres.

Basert på disse tankene endte vi opp med MariaDB Server [34]. MariaDB er en populær database server og ble lagd av de opprinnelige utviklerne av MySQL [35] som er et av verdens mest populære Open Source databaser [36]. Dette var viktig fordi dens relasjon til MySQL gjorde det mulig å bruke 'Go-MySQL-Driver' [37], en driver som ga oss muligheten til å fullstendig benytte oss av Golang-pakken 'database/sql' [38].

6.6 Teknologiskisse

En teknologiskisse viser de forskjellige teknologiene som er brukt i et prosjekt. Vår skisse ligger i figur 6.3. Teknologiene er sortert i fire hovedkategorier; utviklingsmiljø, ressurser, administrativt, og kommunikasjon. Under hver teknologi er en liten beskrivelse om hva den er brukt til.



Figur 6.3: Teknologiskisse med visuelle fremstillinger av teknologiene vi har brukt - Diagram laget ved hjelp av Draw.io

Kapittel 7

Testing

7.1 Brukertester

Målet med testing er å oppdage eventuelle feil, svakheter, sikkerhetshull, og lignende med applikasjonen som utvikles, og i dette kapitlet diskuterer vi testingen vi har gjort for å kvalitetssikre løsningen vår. Vi har fra starten av hatt fokus på å utvikle et brukervennlig program, og ville så tidlig så mulig få tilbakemeldinger på brukergrensesnittet. Derfor begynte vi med ukentlige brukertester allerede i starten av februar. Brukertestene ble i hovedsak gjennomført med oppdragsgiveren og andre kommunearbeidere som er mulige brukere, men vi hadde også brukertester for veilederen og bekjente fra vår egen krets. Sistnevnte var et forsøk på å få tilbakemeldinger fra folk som ikke har vært med på prosjektet, og da med fokus på å sende inn en henvendelse uten å ha kjennskap til prosjektet på forhånd.

Vår oppdragsgiver hadde vanskeligheter med å se for seg løsningen i begynnelsen. Hun visste hva hun ønsket av funksjonalitet, men hadde ikke en formening om hvordan hun ville ha disse i praksis. Det oppdragsgiveren var mest opptatt av gjennom testene og diskusjonene vi hadde rundt design var at kommunen skulle få inn så mange henvendelser som mulig. Nettsiden hvor innbyggere og besøkende sender inn henvendelser, skulle være så enkel at både barn og eldre kunne bruke den. Diskusjonene om designvalg og funksjonalitet som dukket opp under brukertestene var derfor rettet mot hva som ville øke eller minke antall henvendelser sendt inn. Noen av diskusjonene er beskrevet i punkt 7.1.1 til 7.1.4.

7.1.1 Diskusjon: Skal brukerskjemasiden ha bildeeksempler på svartelistede planter Gjøvik er på utkikk etter?

Diskusjonen dukket opp under en demo med flere arbeidere fra kommunen hvor det ble foreslått at brukerskjemasiden burde ha bildeeksempler på de svartelistede plantene man ønsket henvendelser på. Det ble argumentert at mange ikke vet hvordan de typiske svartelistede plantene ser ut, og at det det ville være hjelpsomt å ha et par bilder på brukerskjemasiden som viste de plantene Gjøvik kommune var på utkikk etter. Dette kunne også bidra til at mindre henvendelser av normale planter ble sendt inn.

Motargumentet fra å putte inn slike bilder var at designet mistet sin enkelhet og ble mindre brukervennlig. Oppdragsgiveren vår var bekymret for at folk valgte å ikke sende inn henvendelser når vi gir for mye informasjon på en gang. E.g., brukeren som skal sende inn henvendelse ser bildene og tviler på om de har funnet riktig plante, og velger derfor å ikke sende inn noe.

Til slutt kom vi frem til at det var best å gjøre brukerskjemasiden så enkel så mulig. Det ble derfor bestemt at en vi kan heller lage en separat informasjonsside på kommunens nettsider som gjør innbyggerne kjent med Gjøviks ugress. Merk det ble ikke vår oppgave å legge inn en slik nettside.

7.1.2 Diskusjon: Skal vi la kommunen få bruke innsendte bilder videre?

En gjentagende diskusjon var om bildene som ble sendt inn gjennom brukerskjemasiden kunne bli brukt senere av Gjøvik kommune, e.g., på Facebook-siden deres eller andre nettsider. Dette var et vanskelig spørsmål da bildene faktisk tilhører de som har tatt dem og ikke kommunen. Det var derimot ønskelig å få brukt bildene senere, spesielt visst det var et godt eksempelbilde.

Vi vurderte å ha en avmerkingsboks som sa: «Jeg godtar at Gjøvik kommune kan bruke bildene som er sendt inn» eller noe lignende som gjorde brukeren oppmerksom på at de gir fra seg opphavsrett til bildet. Da kom vi inn på tanken om vi fikk mindre henvendelser når brukeren visste at bildet ble brukt. Det ble bekymring rundt personopplysningsloven [39], da disse bildene kunne inneholde personer og andre private ting, som ikke kunne bli gjort offentlig.

Diskusjonen rund bruk av bilder vekket andre tanker rundt det offentlige kartet, som kunne bli brukt av kommunen for å vise offentligheten dagens fremgang i bekjempelsen av svartelistede planter. Hva hvis kartet inneholdte posisjonsdata fra planter nær private eiendommer? Disse posisjonene ville mest sannsynlig bli oppsøkt av nysgjerrige folk og kunne da bli et brudd på personopplysningsloven. Det ble derfor lagt inn en ekstra avmerkingsboks på administrasjonssidene som markerte om en henvendelse var privat eller offentlig. Merk at det er administratorbrukerens avgjørelse om henvendelsen skal markeres som privat eller offentlig.

7.1.3 Diskusjon: Skal vi kreve at en henvendelse må inneholde et bilde?

Mot slutten av demoene våre dukket opp et spørsmål om vi skal kreve at minst et bilde må bli lagt ved en henvendelse. Uten bilde vil jobben bak å vurdere en henvendelse ta mer tid siden kommunen må fysisk sjekke om posisjonen som henvendelsen viser til har en svartelistet art. Oppdragsgiveren sa at per i dag kunne hun få henvendelser som en enkel SMS uten bilder og det kan godt hende at det vil skje gjennom skjemasiden også. I frykt om at mindre henvendelser ble sendt inn hvis det ble et krav at bilde må legges ved, ble det konkludert at et bilde ikke er nødvendig. Vi informerer ikke brukeren av brukerskjemasiden om at bilde ikke trengs, da ideen kan friste flere i å ikke sende inn et bilde.

7.1.4 Diskusjon: Gammelt vs. nytt design på brukerskjema

Vi hadde på starten av prosjektet laget et brukerskjema som hadde de viktigste elementene for innsending av en svartelistet art. Når vi viste det fram for veileder og oppdragsgiver så var tilbakemeldingene positive. Det var ikke før vi viste fram nettsiden til noen som ikke var relatert til bacheloroppgaven, at vi fikk brukbar kritikk. Det at veileder og oppdragsgiver ikke hadde sterke meninger om det første designet, var at vi hadde allerede jobbet med dem om hva brukerskjemaet burde inneholde. Dermed hadde de visse forventinger og visste allerede hva brukerskjemaet skulle oppnå. Dermed var det lett for dem å forstå hva brukerskjemaet skulle brukes til.

Her er et eksempel på en av brukertestene vi foretok på et par stykker som ikke hadde sett oppgaven før:
Mann 55 år, ble bedt om å fylle inn skjemaet etter at de fant en kjempebjørnekjeks i veikanten. Selve informasjonen ble fylt inn riktig, men stusset litt over kontaktinformasjonen. Spørsmål ble stilt om dette var nødvendig eller pålagt.
Kvinne 31 år, ble bedt om å fylle inn skjemaet etter å ha funnet en plante de tror kan være svartelistet. Litt forvirret når de delte posisjonen sin, men ellers fylte inn informasjon greit. Ble også litt forvirret over kontaktinformasjonen.

Dermed ble brukerskjemaet designet på nytt for å prøve å gjøre det både lettere og mindre forvirrende. Mer detalj om hvilke endringer er beskrevet i punkt 4.1 om design. Det nye designet ble vist fram for både veileder og oppdragsgiver, og ble positivt mottatt. Veileder pekte blant annet ut at bare å ha en knapp 'velg bilder' kan være forvirrende for noen, da hensikten er å ta bilde. Som vist i figur 7.1, så er det dermed to knapper på mobilversjonen. Selv om det er fullt mulig å gå til kameraet ved å klikke på 'velg bilder' og så 'åpne kameraet' på telefonen, er dette et steg ikke alle vet om.

Figur 7.1: PC- og mobilversjon av side nummer to, på det nye brukerskjemaet

Selv om brukertestene hadde vært positive for det nye designet uten 'ta bilde' knappen, viste vi brukerskjemaet på nytt etter å ha lagt inn 'ta bilde' knappen. Responsen var veldig positiv og 100 % av brukerne vi testet på reagerte positivt på denne endringen. Så til tross for at ingen hadde reagert på at man bare kunne velge bilder, så var det fortsatt rom for forbedring.

7.2 Manuell testing

Vi har i tillegg til brukertester, gjennomført en del manuell testing i løpet av oppgaven. Manuell testing krever at det er noe funksjonalitet, så vi begynte ikke med manuell testing før vi hadde fått en fungerende kobling mellom server og frontend.

Når man koder, er det lett å teste for resultatet man er ute etter. Hvis vi ønsker å kode inn at en henvendelse skal forhåndsvise to bilder før man klikker inn på den, så er det naturlig å teste med en henvendelse som har to bilder vedlagt. Det er derimot viktig å teste hva som skjer om man bare legger ved ett bilde, eller fire bilder. I tillegg kan det være at ettersom man må teste flere ganger, så bruker man de samme to bildene gang på gang, som gjør at man ikke tester ulike størrelser på bildene og hva som skjer hvis det ene bildet er tre ganger så stort som det andre.

Her er det dermed viktig å bruke begge gruppemedlemmene mye. Det er ofte ett gruppemedlem som jobber med en del av koden, mens den andre jobber med noe annet. Hvis man da bruker det andre gruppemedlemmet til å teste noe de ikke har kodet like mye selv, vil de teste mer som en vanlig bruker som ikke har like mye formening om hva resultatet burde være. Denne formen for testing kalles kodegjennomgang, som vi gjort gjennom hele oppgaven og har i mange tilfeller funnet en del kritiske feil som ellers ikke ville bli funnet hvis bare den som kodet den delen testet den samme delen.

Ett av mange eksempler på dette, er når vi testet hvordan bildene ble vist fram på administrasjonssidene. Anders hadde testet med en del bilder, hentet fra Google. Alle bildene var nesten kvadratiske, som gjorde at brukergrensesnittet på siden ble designet etter de kvadratiske bildene. Senere skulle Malene teste funksjonaliteten i forhold til bilder, og lastet opp bilder hun hadde tatt på mobilen vertikalt. Resultatet ble overlappende elementer som gjorde siden nær umulig å navigere.

Å opprette nye henvendelser kan fort ta lang tid, da henvendelsene ikke lagret seg før vi hadde en database. For å få en henvendelse, måtte man dermed gå inn på brukerskjemaet for å sende inn en henvendelse før man kunne se den i administrasjonssidene. Dette tok fort veldig lang tid, da under utviklingen så blir endringer gjort som krever omstart på serveren oppimot flere titalls ganger i timen. Dermed lagde vi en egen API som ved et knappetrykk oppretter en henvendelse for bruk i testing. Som vist i figur 7.2, så blir navnet og beskrivelsen satt til tilfeldige kombinasjoner av bokstaver, datoen satt tilfeldig og ellers settes de andre verdiene som standard til en ny henvendelse.

Dette gjorde også framvisningen av programvaren for oppdragsgiver og veileder en del raskere, da vi kunne opprette nye henvendelser for å vise nye funksjoner, med bare ett knappetrykk. Selv om vi ofte hadde forberedt forhåndslagde henvendelser, så skjedde det uforventede feil og kræsjer som gjorde at serveren måtte restartes eller lignende, som igjen gjorde det nyttig å ha en ny henvendelse rett rundt hjørnet.

```
//Sets a numnber of random values
temp := structs.PlantInquiry{
    Name:       "annet..",
    Listname:   userform.CreateId(18),
    Desc:       strings.NewReplacer("d", " ", "f", " ", "a", " ", "m", " ").
    Mail:       "ompom@kmk.com",
    Lng:        "10.68" + strconv.Itoa(rand.Intn(99)) + "123",
    Lat:        "60.78" + strconv.Itoa(rand.Intn(99)) + "379",
    Id:         userform.CreateId(16),
    Date:       ranDate,
    StringDate: ranDate.Format("02. January 2006"),
    Status:     "new",
    Private:    true,
}
database.InsertPlant(temp)
```

Figur 7.2: En del av koden for oppretting av en test henvendelse

Kapittel 8

Installasjon

8.1 Kommersiell løsning

Videre skal vi skrive om installasjonen av løsningen vår. Først vil vi forklare hvordan en kommune kan ta i bruk løsningen, deretter viser vi hvordan vi har installert løsning på våre systemer. Vi ble enig med oppdragsgiver at løsningen burde være kommersielt, altså at vi ikke begrenset løsningen til at bare Gjøvik kommune kunne ta den i bruk. Oppdragsgiver fortalte at kommunene i Norge ikke alltid bruker samme system, e.g., kartsystemet som Gjøvik bruker kan være av en helt annen sort en det Ringsaker kommune bruker. Dette var en av grunnene til at vi ikke implementerte løsningen med Gjøvik kommune sine tjenester og verktøy, men heller bygde den eksternt på 'egen grunn'. Merk at sistnevnte avgjørelse var også preget av andre faktorer som er nevnt i punkt 9.1.1 i avslutningen. Bestemmelsen om å lage en kommersiell løsning ga oss derimot et problem siden vi nå ikke hadde noe spesielt å forholde oss til på teknologifronten; en frihet som var både positiv og vanskelig. Spesielt utfordrende var at vi ikke visste nok om kommunene og deres systemer til at vi kunne ta selvsikre valg på bruk av teknologier. Derfor har vi forsøkt å bruke tjenester vi følte kommuner kunne ta i bruk eller erstatte med sine egne.

Vi har tidligere skrevet om vår bruk av Skyhigh i punkt 6.3 om teknologier, samt om Docker i punkt 6.4 fra samme kapittel. Der snakket vi om hvordan vi har brukt Docker til å implementere løsningen vår i SkyHigh. Vi har Dockerized vår applikasjon for å lette på installasjonsprosessen til kommunene som skulle ta i bruk løsningen. Vi tenkte at kommunene kunne bruke Docker til å installere løsningen slik de ville ha den på sine systemer, e.g., hvor løsningen skulle kjøre, hvilken port den skulle bruke, og andre krav man kan sette for containeren. Dockerfilen vi har laget kan i seg selv endres til kommunens bruk, ved at de kan velge hvilke filer av løsningen de vil ha på serveren sin, men dette forutsier at de vet strukturen på løsningen vår og hvordan de kan dele den opp.

8.2 Database

Databasen som er implementert på SkyHigh er valgt med tanke på at en kommune som Gjøvik kan replisere strukturen. MariaDB er en populær database og vi tenkte dette kunne lette på installasjonen, i.e., de trenger ikke å bruke en spesiell eller høymoderne type database. Teknisk sett trenger ikke databasen å være av typen MariaDB, det eneste kravet vår løsning har er at den må være en relasjonsdatabase (SQL database). Løsningen vår bruker heller ingen tredjeparts løsninger for frontend, noe som definitivt letter på installasjonen hos kommunene som skal ta dette i bruk.

8.3 Sikkerhet

Vi vet derimot at realiseringen av løsningen vår kan by på flere problemer, spesielt ved tanke på sikkerhet. Gjennom vårt arbeid med oppdragsgiveren fikk vi inntrykk av at en kommune har strenge krav til sikkerhet. Det er en stor prosess og integrere nye tjenester inn i deres sikkerhetssystem. Her kommer vi inn på innloggingsdelen av løsningen vår. Ifølge oppdragsgiveren vår bruker ansatte i kommunen én konto for å autentisere seg på flere produkter og tjenester. Det eneste vi visste om disse kontoene var at de brukte brukernavn og passord, derfor implementerte vi en enkel løsning for innloggingen vår som autentiserer administrasjonsbrukere med brukernavn og passord. Merk at innlogging ikke er fullt integrert i vår løsning siden vi fikk knapp med tid. Denne delen ble nedprioritert nettopp fordi kommunen har sin egen måte å integrere eksterne tjenester på, ettersom vår innlogging hadde mest sannsynlig blitt erstattet med deres egen løsning.

8.4 Vår implementasjon

Som nevnt i 6.4 om teknologier så har vi implementert løsningen vår ved bruk av Docker på Skyhigh. Linken til denne implementasjonen er her:
`https://10.212.174.70:8080/henvendelser/`.
Merk at hvis man er utenfor NTNU sitt nettverk, trenger man å koble seg på NTNU sitt nettverk via VPN.

Kapittel 9

Avslutning

9.1 Diskusjoner

Til slutt i rapporten vil vi drøfte og evaluere vårt arbeid og oppgaven vi fikk tildelt, og komme med en konklusjon. Vi starter med å se på noen diskusjoner som har dukket opp under prosjektet vårt og hvordan de påvirket sluttresultatet.

9.1.1 Fra skreddersydd løsning for Gjøvik kommune til en kommersiell løsning

Vi nevnte i kapittel 8 om installasjon at vi hadde planlagt å lage en løsning som var bygget på Gjøvik kommune sine tjenester. Vi tenkte blant annet å bruke deres kartsystem til å lagre posisjonsdata, og deres nettskjema-løsning for å samle inn henvendelser. Vi hadde derfor i begynnelsen flere møter med forskjellige ansatte i Gjøvik kommune som kunne redegjøre hva slags muligheter vi hadde for å bruke deres tjenester. Vi fikk derimot inntrykket av at disse mulighetene var vanskelig å realisere, spesielt med tanke på hvor kort tid vi hadde til disposisjon. I dette underkapitlet kommer vi til å diskutere valgene vi vurderte, og hvordan vi endte opp med å gå for en mer kommersiell løsning.

Vi møtte blant annet med avdelingen for kommunalteknisk drift for å se om vi kunne bruke GPS-systemet de hadde i traktorene som kjørte kantslått. Der fikk vi høre at de ventet på et GPS-system vi sikkert kunne benytte oss av, men denne løsningen ville først bli integrert i deres systemer i april eller mai 2023. Vi innså at å lage et grensesnitt til et GPS-system som ikke var der ble en vanskelig oppgave, og vi måtte gå bort ifra denne planen. Slik løsningen vår er i dag, så kan brukeren av administrasjonssidene opprette en PDF med informasjon om den svartelistede arten som skal bekjempes. Dette ble en slags erstatning for det grensesnittet vi ville utvikle som kunne ha sendt denne informasjonen direkte til GPS-systemet i traktoren som skulle gjennomføre bekjempingen.

Relatert til brukerskjemasiden hadde vi først planlagt å lage et nettskjema gjennom Gjøvik kommune sine nettsider. I forbindelse med dette, hadde vi et møte med en ansatt som var innholdsprodusent og webredaktør i Gjøvik kommune. Dessverre fikk vi fort vite at behovet vårt ble ikke dekket hvis vi bygget nettskjemaet gjennom deres systemer. På baksiden ble nemlig disse nettskjemaene lagret som PDF og håndtert manuelt. Det var derfor ingen mulighet for oss å direkte samle inn dataen fra innsendte henvendelser og behandle de i løsningen vår. Dette var i strid med oppgaven i seg selv, som innebar å lette på prosessen av å hente inn henvendelser om svartelistede arter. Benyttet vi oss av dette nettskjema, måtte noen fysisk lese alle PDF-ene for så å legge inn dataen i løsningen manuelt på noe vis. Dette er ingen forbedring fra hvordan oppdragsgiveren gjør det i dag, hvor hun får henvendelser på epost og gjennom tekstmeldinger for så å føre de inn og behandle de i kommunens journal-system manuelt. Dermed gikk vi bort fra denne planen også og implementerte vår egen brukerskjemaside.

Vi antok at kommunen hadde en kartløsning vi kanskje kunne bruke, derfor hadde vi et møte med to ansatte som jobbet med oppmåling for å se på mulighetene vi hadde. Vi var uheldig i at personen med hovedansvaret for kartløsningen var på permisjon, og vi fikk derfor ikke full oversikt på systemet. På møtet oppdaget vi sammen med oppdragsgiveren at en løsning på hennes problem har blitt prøvd løst et år tidligere. Kommunen hadde skaffet seg en modul for kartsystemet deres siktet til felt, og med denne modulen hadde de lagd et redigerbart kart av Gjøvik hvor oppdragsgiveren kunne legge inn posisjoner til svartelistede plantearter. Det viste seg derimot å bli en komplisert sak å få vår løsning til å kommunisere med denne feltløsning siden det var et lukket brukerprogram i seg selv med ingen muligheter for import eller eksport av posisjonsdata. Med andre ord kunne vi ikke overføre posisjonene vi hentet fra en henvendelse til feltløsningen og motsatt. Igjen stod vi ovenfor en løsning hvor oppdragsgiveren måtte manuelt legge inn data fra et system til et annet. Uten muligheten til å lagre kartet vårt over svartelistede arter med kommunens kartsystem, måtte vi endre på planen vår om å implementere en løsning skreddersydd for Gjøvik kommune.

På dette tidspunktet tok vi kontakt med en annen ansatt fra kommunen som hadde jobbet med NTNU studenter tidligere på et annet prosjekt for å få litt råd om hva vi burde gjøre. Personen anbefalte at vi skulle tenke mer kommersielt og utviklet et system uten å blande inn kommunen sine systemer, i hvert fall ikke for mye. Personen argumenterte at vi ville komme opp med en bedre løsning da dette ville sparte tid og gi oss mer frihet til å utvikle hva vi så for oss. Etter dette møtet ble det en del endringer i prosjektplanen vår, men disse lettet på oppgaven betraktelig og vi fikk motivasjon til å lage en løsning som kunne hjelpe flere kommuner med samme problem som Gjøvik.

9.1.2 Effektiv databasehåndtering

I punkt 5.3.3 om implementering av databasen diskuterte vi hvorvidt det var lønnsomt å stadig sende forespørsler til databasen under programkjøringen. Ved listing av alle henvendelser, en operasjon som skjer ofte i løpet av programkjøringen, blir det gjort en stor forespørsel til databasen som henter alle henvendelser. Denne operasjonen er ikke effektiv og kan risikere å gjøre programmet tregt å bruke. Vi hadde som mål å lage et brukervennlig og effektivt program, men med denne implementeringen er effektiviteten like bra som vi hadde ønsket. Vi kunne for eksempel implementert en buffer som midlertidig lagrer henvendelser lokalt så vi slipper så å hente alle henvendelsene fra databasen hver gang de etterspørres. Bufferen fylles opp ved programstart, og oppdateres bare med henvendelser som nylig har blitt endret på i databasen.

9.2 Kritikk av oppgaven

Videre vil vi kritisere oppgaven vi fikk. Selv om bacheloroppgaven har blitt vurdert som bra nok, er det vanskelig å forutsi nøyaktig alle aspekter ved en bacheloroppgave, som kan føre til at vi må tilpasse oss.

9.2.1 Samarbeid med kommunen

Vi hadde et godt samarbeid med oppdragsgiveren og kommunen, men det dukket likevel opp problemer underveis som påvirket oppgaven. Blant annet involverte oppgaven vår flere avdelinger i kommunen enn bare landbruksavdelingen som var avdelingen til oppdragsgiveren vår. Vi strevde derfor litt i begynnelsen med å diskutere mulighetene og tankene våre på tvers av avdelingene. Vi kom litt sent i gang med oppgaven som et resultat av alle møtene vi hadde med avdelingene, spredt over flere uker. Et av våre siste møter i denne perioden var med noen som hadde jobbet med NTNU studenter før (se punkt 9.1.1), og vi angret på at vi ikke møtte med denne personen først. Vi kunne ha spart oss mye tid hadde vi blitt forklart fra begynnelsen av at det best å utvikle løsningen eksternt på 'egen grunn'.

En annen faktor som påvirket oppgaven, var da vi oppdaget feltløsningen Gjøvik kommune hadde lagd med sitt kartsystem for å løse problemet til oppdragsgiveren vår et år tidligere (igjen se punkt 9.1.1). Oppdragsgiveren visste ikke at en slik løsning hadde blitt lagd, og elsket ideen da vi kikket på funksjonalitetene til denne løsningen. Vi følte at oppgaven vår allerede var løst og mistet en del motivasjon for bacheloroppgaven. Vi lurte på hva vi skulle gjøre nå for å fortsatt ha en oppgave å utvikle. Deres feltløsning dekket derimot ikke alt behovet til oppdragsgiveren så vi, e.g., innsending og behandlingen av henvendelser, noe som ga oss nytt mot til å fortsette.

9.3 Videre arbeid

Bacheloroppgaven har en lengde på cirka fire måneder, som ikke er veldig lang tid til å utvikle noe nytt. Vi hadde en rekke ideer og ønsker om hva nettsidene kunne blitt, hadde vi hatt flere ressurser og mer tid.

9.3.1 Kunstig intelligens

Et av de første ideene vi kom på sammen med veileder, var bruk av kunstig intelligens. Mens denne bacheloroppgaven gjennomføres i starten av 2023, har det vært et stort sprang i bruk av kunstig intelligens (KI), som for eksempel bruken av ChatGPT [40]. Dermed kom ideen om bruk av KI, da det finnes allerede apper som gjenkjenner planter via bilder. For eksempel, kan man bruke Google Lens, Bing [41] eller artsorakelet [42].

Ideen vår var dermed å integrere en av disse løsningene med brukerskjemaet vårt. Noen planter kan ligne på hverandre, som gjør det vanskelig for mindre erfarne planteentusiaster til å gjenkjenne alle plantene som er svartelistet. Når man da åpner brukerskjemaet og tar bildet av planten, vil KI prøve å identifisere planten før den sendes inn. Finner KI ut av at planten ikke er svartelistet, kan man gi beskjed til bruker om at det ikke er behov for å sende inn skjemaet allikevel. I tillegg kan man også gjøre det motsatte, hjelpe bruker med å gjenkjenne en svartelistet plante hvis de ikke er sikre på om den er svartelistet eller ikke. Dette ville redusert henvendelser som ikke er relevante og muligens økt antall henvendelser som faktisk har svartelistede planter.

9.3.2 Samarbeid med andre systemer

Ettersom kommunalteknisk drift ikke hadde et GPS-system på plass, gikk vi aldri videre med å koble opp vårt system med deres. Hadde vi hatt lengre tid, kunne vi ha planlagt å integrere oss med dem, mens de setter opp sitt system. Dermed kunne vi ha automatisert prosessen med å sette henvendelser til 'under bekjempning' og 'ferdig bekjempet' en del mer enn den er i dag. Vi kunne for eksempel satt det opp slik at alle 'godkjente' henvendelser gjøres tilgjengelig for kommunalteknisk drift, slik at når de skal ut å kantklippe, så henter de bare ut de godkjente henvendelsene som er i ruten de har planlagt å klippe. Når jobben er gjort, kan da sette de henvendelsene som er bekjempet til 'ferdig bekjempet'.

Det er ikke bare kantklipping som brukes for bekjemping av svartelistede planter, noen ganger blir det gjort dugnader og et mannskap går rundt og fjerner planter litt mer manuelt, ettersom ikke alle plantene helt langs veikantene. Dermed vil PDF løsningen vår fortsatt fungere utmerket for de tilfellene det trengs. Så en slags hybridløsning av automatisert kantklipping og PDF ville vært en god ide å prøve ut hadde vi hatt enda lenger tid på oppgaven.

9.3.3 Tilgjengelighet

I tillegg til ekstra funksjonalitet, er ekstra tilgjengelighet like viktig når det kommer til nettsider. Det finnes en rekke retningslinjer når det kommer til design av nettsider. Dette gjelder blant annet kontrast, skriftstørrelse, tekst for bilde, med mer. Disse retningslinjene kalles 'Web Content Accessibility Guidelines' [43], som inneholder alt av diverse måter å designe nettsidene sine på slik at de er tilgjengelig for alle. Selv om mange av grunnprinsippene er møtt, er det mange som ikke er møtt. For eksempel har vi ikke gjort nettsidene tilgjengelig på engelsk, og vi har ikke testet nettsidene med bruk av tekst-til-lyd dersom svaksynte eller blinde skal bruke nettsiden. Å bearbeide alle retningslinjer tar lang tid, da det er mange aspekter man skal ta hensyn til. så fire måneder til å legge en plan, kode fra bunnen av og i tillegg skrive en rapport, så blir det for lite tid til å inkludere alle retningslinjene. Dermed hadde dette vært et utmerket punkt å muligens jobbe videre med i fremtiden.

9.4 Evaluering av gruppen

Gruppen har samarbeidet veldig godt. Ettersom det bare er to av oss, er det lett å følge med på hva det andre medlemmet har gjort. Dermed har samarbeid vært sterkt og vi har fulgt hverandre opp hele veien. Vi har brukt Git mye for å fordele arbeid, og ettersom vi har såpass motsatte preferanser når det kommer til selve programmeringsdelen, så har vi hatt stor fordel av det.

I forhold til skriving av rapport og prosjektplan, så har vi fulgt hverandre tett og skrevet mye sammen, slik at begge sine tanker og meninger kom fram. Valget av utviklingsmetodikk som Scrum, ser vi i ettertid var et godt valg. Scrum tillot oss å jobbe hver for oss, men samtidig ha et tett samarbeid hvor vi har hatt jevnlige møter og en god arbeidsflyt helt fra starten av.

9.5 Konklusjon

Vi har lagd en nettsideløsning for Gjøvik kommune som systematiserer bekjempelsen av svartelistede arter i veikanten. Når vi ser tilbake på prosjektplanen vår, ser vi at vi har nådd de fleste prosjektmålene vi satte oss. De oppnådde effektmålene omhandlet å samle arbeidsområdet til oppdragsgiveren på én plattform, og redusere tiden og mengden arbeid det tar å behandle en henvendelse. Læringsmålene vi har oppnådd var å utvikle et prosjekt hvor vi måtte forholde oss til en oppdragsgiver med et reelt problem, og ikke minst anvende kunnskap fra tidligere emner i studiet, samt bruke nye og kjente verktøy. Vi har hatt god bruk av verktøykassen vår med ferdigheter og kompetanse som vi har bygget opp gjennom studiet. Vi har også oppnådd resultatmålene våre som var å lage to fungerende nettsideløsninger som omhandler bekjempingen av svartelistede planter. Prosjektet har også bidratt til en bevisstgjøring hos Gjøvik kommune og dens forskjellige avdelinger om hva som har blitt gjort tidligere og hva som ikke har blitt fullført angående deres optimaliseringsplan på bekjempelsen av svartelistede plantearter.

Bibliografi

[1] «How many plants are in the world?» (2023), adresse: https://a-z-animals.com/blog/how-many-plants-are-in-the-world/ (sjekket 28.01.2023).

[2] «How many plants there are in the world and how they're doing.» (2016), adresse: https://www.lifegate.com/study-world-plants (sjekket 28.01.2023).

[3] «Cultivated plant globalization.» (2019), adresse: https://plantspeopleplanet.au/o1/l25/ (sjekket 30.01.2023).

[4] «Svartelista"– et begrep som ble brukt frem til 2018.» (2018), adresse: https://artsdatabanken.no/Pages/233516/_Svartelista____et_begrep_som (sjekket 16.01.2023).

[5] «Kjempebjørnekjeks – staselig, skadelig og fremmed.» (2020), adresse: https://www.nibio.no/nyheter/kjempebjornekjeks--staselig-skadelig-og-fremmed (sjekket 30.01.2023).

[6] «Fremmedartslista.» (2023), adresse: https://snl.no/Fremmedartslista (sjekket 26.01.2023).

[7] «PROG2005 - Cloud Technologies.» (2023), adresse: https://www.ntnu.no/studier/emner/PROG2005/#tab=omEmnet (sjekket 06.05.2023).

[8] «PROG2006 - Avansert programmering.» (2023), adresse: https://www.ntnu.no/studier/emner/PROG2006/#tab=omEmnet (sjekket 06.05.2023).

[9] «IDATG2204 - Datamodellering og databasesystemer.» (2023), adresse: https://www.ntnu.no/studier/emner/IDATG2204/#tab=omEmnet (sjekket 06.05.2023).

[10] «PROG2053 - Webteknologier.» (2023), adresse: https://www.ntnu.no/studier/emner/PROG2053/#tab=omEmnet (sjekket 06.05.2023).

[11] «PROG2052 - Integrasjonsprosjekt.» (2023), adresse: https://www.ntnu.no/studier/emner/PROG2052/#tab=omEmnet (sjekket 06.05.2023).

[12] «VisBook.» (2023), adresse: https://visbook.com/no/ (sjekket 07.05.2023).

[13] «What is scrum?» (), adresse: https://www.scrum.org/resources/what-is-scrum (sjekket 25.01.2023).

[14] «What Is Extreme Programming (XP)? and It's Values, Principles, And Practices.» (), adresse: https://www.nimblework.com/agile/extreme-programming-xp/ (sjekket 08.05.2023).

[15] «Kanban.» (), adresse: https://www.agilealliance.org/glossary/kanban/ (sjekket 08.05.2023).

[16] «What is Agile?» (), adresse: https://www.atlassian.com/agile (sjekket 08.05.2023).

[17] «image metadata.» (2023), adresse: https://www.techtarget.com/whatis/definition/image-metadata (sjekket 04.05.2023).

[18] «Base64.» (2023), adresse: https://developer.mozilla.org/en-US/docs/Glossary/Base64 (sjekket 23.04.2023).

[19] «Protect your privacy and turn geotagging off your smartphone photos.» (2022), adresse: https://blog.credo.com/2022/04/28/protect-your-privacy-and-turn-geotagging-off-your-smartphone-photos/ (sjekket 24.04.2023).

[20] «Hvordan bruke verktøy for nettleserutvikler.» (2020), adresse: https://hvordan-apne.com/blog/hvordan-bruke-verktoy-for-nettleserutvikler/ (sjekket 24.04.2023).

[21] «Golang.» (2023), adresse: https://go.dev/ (sjekket 15.04.2023).

[22] «Which Programming Languages Use the Least Electricity?» (2018), adresse: https://thenewstack.io/which-programming-languages-use-the-least-electricity/ (sjekket 01.05.2023).

[23] «Why Golang Is So Fast: A Performance Analysis.» (), adresse: https://www.bairesdev.com/blog/why-golang-is-so-fast-performance-analysis/ (sjekket 01.05.2023).

[24] «Python.» (2023), adresse: https://www.python.org/ (sjekket 10.05.2023).

[25] «C++.» (2023), adresse: https://cplusplus.com/ (sjekket 10.05.2023).

[26] «Java Programming Language.» (2023), adresse: https://docs.oracle.com/javase/8/docs/technotes/guides/language/index.html (sjekket 10.05.2023).

[27] «Golang pakke - HTML/template.» (2023), adresse: https://pkg.go.dev/html/template (sjekket 20.04.2023).

[28] «What's the fastest template engine in Go?» (2020), adresse: https://medium.com/@kataras/whats-the-fastest-template-engine-in-go-fdf0cb95899b (sjekket 28.04.2023).

[29] «Openstack at NTNU.» (2023), adresse: https://www.ntnu.no/wiki/display/skyhigh (sjekket 10.04.2023).

[30] «Certificates for localhost.» (2017), adresse: https://letsencrypt.org/docs/certificates-for-localhost/ (sjekket 29.03.2023).

[31] «SSL Certificate for Localhost — Is It Possible?» (2023), adresse: https://comodosslstore.com/resources/ssl-certificate-for-localhost/ (sjekket 10.04.2023).

[32] «Docker.» (2023), adresse: https://www.docker.com/ (sjekket 01.05.2023).

[33] «CockroachDB.» (2023), adresse: https://www.cockroachlabs.com/ (sjekket 04.05.2023).

[34] «MariaDB Server.» (2023), adresse: https://mariadb.org/ (sjekket 04.05.2023).

[35] «MariaDB in brief.» (2023), adresse: https://mariadb.org/en/ (sjekket 04.05.2023).

[36] «What is MySQL?» (2023), adresse: https://www.oracle.com/mysql/what-is-mysql/ (sjekket 04.05.2023).

[37] «Database driver - Go-MySQL-Driver.» (2023), adresse: https://github.com/go-sql-driver/mysql/ (sjekket 04.05.2023).

[38] «Golang pakke - database/sql.» (2023), adresse: https://pkg.go.dev/database/sql (sjekket 04.05.2023).

[39] «Lov om behandling av personopplysninger (personopplysningsloven).» (2022), adresse: https://lovdata.no/dokument/NL/lov/2018-06-15-38 (sjekket 16.05.2023).

[40] «Number of ChatGPT Users (2023).» (2023), adresse: https://explodingtopics.com/blog/chatgpt-users (sjekket 18.05.2023).

[41] «The 5 Best Plant Identifier Apps for Android and iPhone.» (2022), adresse: https://www.makeuseof.com/tag/identify-plants-flowers-phone-camera/ (sjekket 14.05.2023).

[42] «Ny app lar deg sjekke norske dyre- og blomsterarter.» (2020), adresse: https://dinside.dagbladet.no/fritid/ny-app-lar-deg-sjekke-norske-dyre--og-blomsterarter/72677067 (sjekket 14.05.2023).

[43] «WCAG 2 Overview.» (2023), adresse: https://www.w3.org/WAI/standards-guidelines/wcag/ (sjekket 15.05.2023).

[44] «Fremmede arter.» (2023), adresse: https://www.gjovik.kommune.no/tjenester/naring-og-landbruk/landbruk/jordbruk/fremmede-arter-2/ (sjekket 26.01.2023).

Vedlegg A

Prosjektplan

Bacheloroppgave PROG2900 - Prosjektplan Svartelistede arter i vegkanten

Deltakere
Malene Lundemo
Anders Lunde Hagen 1. februar, 2023

A.1 Mål og rammer

A.1.1 Bakgrunn

Verden består av mange forskjellige plantearter spredt rundt omkring hele jordkloden [1][2]. Noen arter lever i symbiose, mens andre arter er mer selvstendige. For å overleve, har noen planter utviklet seg ekstra høye for å skaffe seg mer sollys, mens noen andre planter har gjort seg ekstra fargerike for å sørge for at pollen blir spredt ved hjelp av insekter. Det er en konstant kamp mellom plantene om hvem som overlever, og noen ganger så blir en plante så utkonkurrert at de ikke lenger klarer å overleve eller spre seg, som gjør at plantearten til slutt dør ut.
Noen steder kan det være mange planter, som gjør at plantene må være flinke til å spre seg fort, gjøre seg stor eller overleve på lite næring. Dersom disse plantene sprer seg til et sted hvor det ikke er mye motstand fra andre planter, kan det være at de tar helt over og utkonkurrerer andre plantearter. Dette kan skje naturlig, men det kan også skje som følge av globalisering [3]. Dersom en plante blir introdusert i et nytt område, kan det være at den blir svartelistet i det området.

I Norge har vi en rekke lister over diverse planter og arter på artsdatabanken.no. Her er det blant annet en liste over svartelistede arter som utgjør en risiko for norsk natur. Svartelistede arter er et eldre begrep som har blitt endret til fremmed arter"[4], men ettersom oppgavetittelen inneholder ordet svartelistedeså vil vi også bruke dette begrepet fremover.
Et eksempel på en svartelistet art er kjempebjørnekjeksen. Opprinnelig fra de vestlige delene av Russland, men ble introdusert til andre deler i Europa av mennesker på 1800-tallet [5][6]. Kjempebjørnekjeksen er et godt eksempel på en plante som andre planter i Norge ikke har lært seg å konkurrere imot. Den er stor og høy, med store blader, som skygger solen for andre planter. Den sprer seg lett, da hver plante produserer titusenvis av frø. I tillegg er den giftig for mennesker, da plantesaften kan gi utslett på huden. Mye av ansvaret i Norge ligger på et kommunalt nivå når det kommer til bekjemping av svartelistede arter.

A.1.2 Prosjektmål

Resultatmål

Vårt mål er å lage to nettsideløsninger som kan brukes både på mobil og PC, med hver sin oppgave. Den ene nettsideløsningen skal fungere som et innsendingsskjema hvor personer i Gjøvik kommune kan sende inn mulige funn av svartelistede arter til kommunen for gjennomgang.
Den andre nettsideløsningen vil kunne motta alle henvendelser og behandle de, i tillegg skal man kunne registrere eventuelle nye artsfunn til artsdatabanken.no. GPS-koordinater skal kunne ekporteres fra nettsiden og deretter importeres i kommunens kartsystemer, og da bli tilgjenglig for de som har ansvar for å fjerne artene.

Effektmål

Det er ingen standardisert løsning for å sende inn henvendelser til kommunen i dag. Hele prosessen fra henvendelsen blir sendt inn, til traktorføreren får beskjed om utkjøring, er tungvinn da alt må gjøres manuelt. Vi ønsker dermed å lette på arbeidsmengden, ved å standardisere og automatisere. Mer konkret, ønsker vi å nå effektmålene:

- Redusere antall platformer/verktøy brukt for henvendelser angående svartelistede arter, da det sendes på alt fra SMS, epost til Facebook meldinger.
- Redusere tiden brukt fra en innbygger sender inn en henvendelse til traktorføreren får beskjed om hvor de skal kjøre ut.
- Øke antall henvendelser fra innbyggere, ved å gjøre det lettere å vite hvor man skal henvende seg ved bruk av løsningen. Målet er å øke dagens antall henvendelser til landbruksavdelingen fra 10 til 20 per måned.
- Redusere mengden arbeid som må bli gjort per henvendelse.
- Føre automatisk statistikk på plantearter

Læringsmål

- Hvordan jobbe som utviklere med en oppdragsgiver.
- Lære hvordan man skal gå fram for å løse en oppgave som ikke har spesifisert alle aspektene.
- Bli bedre på å anvende relevante verktøy i forhold til oppgaven, som Gitlab, epost, Teams og lignende.
- Lære oss å tilpasse oss ettersom hva som må prioriteres, med tanke på at vi har både en oppgave og en rapport som skal jobbes på.
- Anvende kunnskap fra tidligere emner i studiet, til å forstå hvilke prioriteringer vi burde gjøre og hvordan vi kan anvende verktøy i best mulig grad.

A.1.3 Rammer

- Vi har en begrenset tidsramme på 4 måneder, hvor oppgaven og rapporten skal være ferdig levert innen 22. mai 2023.
- Nettsiden skal fungere på alle platformer.
- Henvendelser og kartdata skal kunne lagres på servere eid av Gjøvik kommune.

A.2 Omfang

A.2.1 Fagområde

Gjøvik kommune har de siste årene hatt en økning i svartelistede arter, og da særlig planter. Svartelistede arter er arter som finnes utenfor sitt naturlige utbredelsesområde og spredningspotensiale. Disse artene blir derfor en stor trussel mot naturmangfoldet da de kan utrydde stedegne arter [44]. Herunder, vil svartelistede arter referere til svartelistede plantearter.

Ansvaret ligger hos kommunen å opprettholde naturmangfoldet i Gjøvik, men det er en vanskelig oppgave ettersom arealet er såpass stort. Innsatsen fra kommunens ansatte strekker ikke til i å oppdage og bekjempe alle svartelistede arter, dermed er det svært verdifullt at innbyggere og besøkende sender inn henvendelser om funn av slike arter. Henvendelsene bidrar også til at den svartelistede arten blir oppdaget tidlig, i.e., før den eventuelt sprer seg over et større område, noe som igjen vil gjøre selve bekjempelsen lettere.

A.2.2 Avgrensning

Løsningen vår skal kunne kommunisere med eksisterende kartløsninger som brukes i de fleste kommuner i Norge. Vi skal lage et grensesnitt som gjør at en fra kommunen kan utnytte de kartlagte svartelistede artene i sine kartsystem. Oppgaven avgrenses derfor til at vi ikke skal lage en skreddersydd løsning for Gjøvik kommune, men da et mer kommersielt fokus.
Kjennskap om svartelistede arter og statistikk om naturmangfoldet i Gjøvik er ikke vår oppgave, denne kunnskapen ligger hos Gjøvik kommune, i.e., i dette tilfellet landbrukssjefen. Alle henvendelser om svartelistede arter skal derfor kontrolleres manuelt i vår løsning, i.e., brukeren har selv ansvar for at henvendelsen blir vurdert korrekt.

A.2.3 Oppgavebeskrivelse

Vår oppgave er å forenkle registreringen og dermed bekjempelsen av svartelistede arter i Gjøvik kommune. Dette er en trinnvis oppgave som vi mener trenger to separate løsninger som kommuniserer med hverandre.
Første trinn ligger hos en tilfeldigperson i Gjøvik som ønsker å varsle kommunen om en eventuell svartelistet art.
Andre trinn er å behandle denne henvendelsen og registrere den mulige svartelistede arten, i.e., kartlegge den.
Tredje og siste trinn er å redusere bestanden til de svartelistede artene som er kartlagt.

Løsning 1 har som mål å dekke første trinn nevnt ovenfor. Dette blir en nettside som er tilgjengelig for alle aktive innbyggerne og besøkende i Gjøvik kommune som oppdager en plante de tror er svartelistet. Nettsiden skal ha disse funksjonalitetene (merk at brukeren i punktene under kan være hvem som helst i Gjøvik som har mulighet til å besøke nettsiden):

- Nettsiden må kunne besøkes gjennom mobilen for å gjøre den lett å bruke.
- Brukeren skal kunne sende inn et bilde av planten de tror er svartelistet/-farlig, samt en liten beskrivelse, e.g., detaljer eller spørsmål.
- Brukeren må få en tilbakemelding om at henvendelsen er tatt imot og eventuelt en til når planten er fjernet.
- Nettsiden må for hver henvendelse hente inn posisjonsdata om hvor planten befinner seg.

Løsning 2 skal dekke de to siste trinnene nevnt ovenfor. Henvendelser fra løsning 1 skal bli sendt til løsning 2 for behandling. Behandlingen inngår å vurdere den mulige svartelistede arten og kartlegge den. Dette vil foregå gjennom en nettside. Funksjonalitetene til denne nettsiden er som følger (merk brukeren i dette tilfellet er en ansatt i Gjøvik kommune med spesifikk tilgang):

- Brukeren må ha brukernavn og passord.
- Den skal være enkel å navigere og bruke.
- Den skal holde oversikt over alle nye og tidligere innsendte henvendelser fra aktive personer i Gjøvik kommune (se løsning 1).
- Brukeren skal kunne markere at plantearten som er henvendt er farlig eller ufarlig, i.e., registrere at dette er en svartelistet planteart.
- Registreringen i punktet over vil føre til at plantens posisjon kartlegges. Målet er at videre skal traktorførere og andre arbeidere som bekjemper plantene kunne benytte seg av samme posisjonsdata.
- Brukeren skal kunne se, endre, fjerne og legge til posisjonsdata for de svartelistede artene.
- Brukeren må ha oversikt over historikk av bekjempelsene som er gjort eller skal skje.
- Kartleggingen av de svartelistede artene skal kunne eksporteres og lagres i kommunens kartsystem.
- Det skal gjennom nettsiden kunne registreres nye plantearter i artsdatabanken.

A.3 Prosjektorganisering

A.3.1 Ansvarsforhold og roller

Scrum teamet består av produkteier Gjøvik kommune, scrum master Malene Lundemo, og utvikler Anders Lunde Hagen.
Produkteieren skal representere hva Gjøvik kommune ønsker ut av prosjektet og hvilken løsning de ser for seg å bruke. Produkteier er også vår oppdragsgiver, og er i denne rapporten omtalt som oppdragsgiver.
Scrum master skal sørge for at utviklingen går i riktig retning i forhold til ønskene til oppdragsgiveren, samt etter mål og rammer. Personen har også ansvaret for sprint review og scrum møter. Merk at i dette prosjektet vil scrum master også fungere som prosjektleder, og har derfor et ansvar å løse eventuelle uenigheter og konflikter i gruppen.
Ettersom gruppen består av to gruppemedlemmer, er begge ansvarlig for å utvikle og gjennomføre de nødvendige oppgavene under hele prosjektet.

A.3.2 Rutiner og grupperegler

Grupperegler

1. Hvert gruppemedlem skal jobbe minst 30 timer i uken, hvor det hovedsakelig arbeides alle hverdagene. Gruppen skal helst møtes 2-3 ganger fysisk per uke.
2. Arbeidstimene skal loggføres med fargekoder basert på hva som er gjort.
3. Det skal gis beskjed minst 1 time på forhånd om en ikke kan møte opp til avtalt tid, spesielt ved scrum møter, veiledningstimer, og møte med oppdragsgiver.
4. Hvis et gruppemedlem ikke klarer å følge arbeidsreglene, eller at det oppstår uenigheter eller konflikter i gruppen, skal rutiner ved regelbrudd følges.

Arbeidsregler

1. Det er møteplikt på scrum møter, veiledningstimer, og møte med oppdragsgiver, samt andre avtalte møter i gruppen.
2. Gruppen forventes å delta i arbeid og oppgaver på eget initiativ.
3. Dersom et gruppemedlem ikke er fornøyd med en annens innsats skal det gis beskjed tidlig.
4. Det skal skrives møtereferat etter alle møter.
5. Dersom et gruppemedlem med god grunn ikke kan møte til avtalte gruppemøter må møtet flyttes til en annen dag. Dersom dette møtet er med veilederen eller oppdragsgiveren må møtet fortsatt gjennomføres med det ene medlemmet.
6. Det forventes at hvert gruppemedlem følger standarder ved koding og verktøybruk.
7. Dersom noe faglig er uforståelig skal gruppemedlemmet tilegne seg kunnskapen som trengs på eget initiativ.

Rutiner ved regelbrudd

1. Et gruppemøte om problemet skal holdes hvor problemet blir forsøkt løst med muntlig diskusjon.
2. Skriftlig advarsel hvor vedkommende skal informeres om:
   a. Hvilket regelbrudd handler det om.
   b. Hva som kreves for å oppveie for tapt arbeid.
   c. Konsekvensene for videre brudd på reglene.
3. Et møte med gruppen og veilederen skal holdes om problemet.
4. Skriftlig ekskludering fra gruppen signert av den andre og emneansvarlig må kontaktes umiddelbart.

A.4 Planlegging

A.4.1 Prosessrammeverk

For valg av prosessrammeverk, vil vi se på hvordan de neste månedene kommer til å bli bygd opp. Vi vil ha ukentlige møter med veileder, i tillegg til at det er satt opp tid til ukentlige møter med oppdragsgiver. Et rammeverk som setter søkelys på ukentlige progresjon vil dermed stemme godt overens med de ukentlige møtene vi har.

Scrum [13] vil dermed være et godt valg av rammeverk. Begge gruppemedlemmene har i tillegg litt erfaring med scrum fra et tidligere prosjekt i studiet. Andre rammeverk, som Kanban vil også være relevante, strukturen med å sette opp to-do-lister kan være nyttig for oss. Derimot er vi bare to gruppemedlemmer, så vi har ikke like stor nytte av en to-do-liste som en større gruppe. I tillegg er scrum mer tidsorientert rundt ukentlige møter som passer godt i forhold til at vi har en begrenset oppgaveperiode på cirka fire måneder.

Ettersom gruppemedlemmene har hatt scrum før, er det lettere å argumentere for at vi skal bruke samme rammeverk igjen. Den største fordelen med scrum er at det forventes en viss mengde arbeid hver uke. Det er ofte lettere å planlegge på forhånd hva som må gjøres, som gjør scrum til et godt rammeverk for oss.

A.4.2 Plan for statusmøter

Vi har et fast ukentlig opplegg som vi bygger på. På fredager skal det holdes sprint review møte etterfulgt av et scrum planning møte på mandagen. fredagsmøtene kan være fysisk og digitalt, men begge gruppemedlemmer har møteplikt. Onsdager er det fysisk møte med veileder, Peter Nussbaum. Møte med oppdragsgiver, Ingun Revhaug, er det satt av tid til møte en gang i uken, men dersom det ikke er nødvendig og møtes hver uke, skal det ikke gå lengre enn to uker mellom møtene. Bruk av plattformene Outlook, Microsoft Teams, og Discord skal aktivt brukes under dette prosjektet, og det forventes derfor at gruppemedlemmene er pålogget og tilgjengelig, i hvert fall i hverdagene.

A.5 Organisering av kvalitetssikring

A.5.1 Dokumentasjon, standarder og verktøy

Dokumentasjon
Dokumentasjon skal jevnlig gjøres mens oppgaven utføres. Det er lett å glemme detaljer i ettertid, så vi skal opprettholde jevnlig dokumentering i hele perioden. Møtereferat og møtenotater blir lagt inn i en delt gruppe på Discord etter hvert møte. Rapporten vil bli skrevet i Overleaf, som bruker LaTeX.

I koden skal vi legge inn kommentarer på hver funksjon som beskriver hva funksjonen gjør. Det vil i tillegg skrives en readmefil som forklarer hva koden gjør som en slags manual.

Standarder
Kildekoden vår vil følge standarder relevante for det språket som blir brukt. For backend vil vi følge standardene til Golang, og frontend vil vi følge standardene til Javascript. Vi vil også følge standarder, som kommentering av funksjoner og klasser, og hyppige commits med bruk av branches når vi koder.

Vi vil ha en main branch hvor den siste stabile versjonen av programvaren ligger, som bare vil bli pushet til etter at ny kode har blitt ordentlig testet for feil. Nye brancher vil bli opprettet hyppig for hver del av koden som skal kodes og bli slettet når vi er ferdig for å opprettholde et ryddig Git repo. Brancher som skal flettes inn i main, vil først bli testet og godkjent av begge gruppemedlemmene før den blir pushet i main.

Utviklingsrutiner

- Git branches skal brukes og skal sjekkes før de merges til master.
- Alle commit beskjeder skal være forståelig og informativ.
- Hver merge request skal inneholde en issue ID til en GitLab issue. Issuen må da beskrive hva som er gjort og eventuelt hva som burde vurderes før den merges inn i main. Dette er spesielt viktig for branches som inneholder bugfix.
- Issuer skal sorteres ved hjelp av Labels, i.e., backlog, påbegynt, review, testing, bugfix.

Verktøy

	Beskrivelse	Brukes til
Gitlab	Versjonskontrollverktøy for koding tilhørende NTNU	Felles lagring av kildekode, backlog og issues
TeamGantt	En nettside hvor man kan lage gantt diagrammer	Oppretting av gantt diagrammer
Microsoft Teams	Kommunikasjonsplattform for samtaler og delte dokumenter	Timeliste og online møter med oppdragsgiver
Discord	Kommunikasjonsplattform for samtaler og tekst	Planlegging og kommunikasjon mellom gruppemedlemmene
Overleaf	Skybasert redigeringsverktøy i LaTeX	Skrive prosjektplan og rapport

Tabell A.1: Navn på verktøy og beskrivelse

A.5.2 Testing

En av hovedaspektene ved denne oppgaven, er å gjøre det lettere for brukere å kunne sende inn henvendelser, i tillegg til at de i kommunen skal ha en mindre tungvinn jobb. Derfor vil vi aktivt bruke brukertesting for å få et bedre innblikk i hva brukere av systemet foretrekker av diverse valg vi gjør i løpet av prosessen. I tillegg vil målgruppen til programvaren være alle aldre, så det er viktig at vi tester med ulike aldersgrupper for å passe på at programvaren er lett å forstå og bruke for alle som eventuelt vil bruke den.

A.5.3 Risikoanalyse

Vi har vurdert 10 risikoer i dette prosjektet og disse er analysert i Tabell 2. Hver risiko har en sannsynlighet for at den inntreffer og eventuelle konsekvenser. Vi har vurdert sannsynligheten binært, i.e., sannsynlig eller usannsynlig. Konsekvensen av en risiko er enten uproblematisk, problematisk, eller kritisk, hvor kritisk regnes som høyeste konsekvens. Der vi har et tiltak mot risikoen, vil tabellen ha et 'Ja' i kolonnen 'Tiltak', hvis vi ikke har et tiltak, vil det stå 'Nei' i denne kolonnen. Med tiltak menes det at vi har planlagt å gjøre noe aktivt for å minske sannsynligheten eller konsekvensen for at risikoen finner sted. Hva de forskjellige tiltakene innebærer er beskrevet i Tabell 3.

	Risiko	Sannsynlighet	Konsekvens	Tiltak
1	Store endringer i kravspesifisering fra oppdragsgiver underveis i prosjektperioden	Sannsynlig	Problematisk	Ja
2	Et av gruppemedlemmene blir syke eller fraværende over en lengre periode	Usannsynlig	Kritisk	Nei
3	Tap av kildekode eller dokumentasjon	Usannsynlig	Kritisk	Ja
4	Teknologier viser seg å ikke være kompatible med hverandre	Usannsynlig	Problematisk	Ja
5	Manglende kompetanse	Sannsynlig	Problematisk	Ja
6	Konkurrenter har et lignende produkt	Sannsynlig	Uproblematisk	Nei
7	Oppdragsgiver er ikke til stede eller ikke mulig å få tak i	Usannsynlig	Kritisk	Nei
8	Veileder er ikke til stede eller ikke mulig å få tak i	Usannsynlig	Problematisk	Nei
9	Uenigheter i gruppen eller ulik arbeidsinnsats	Usannsynlig	Problematisk	Ja
10	Lagring av data kan gå under personvernlover.	Usannsynlig	Problematisk	Ja

Tabell A.2: Risikoanalyse

A.5.4 Tiltak for risikoer

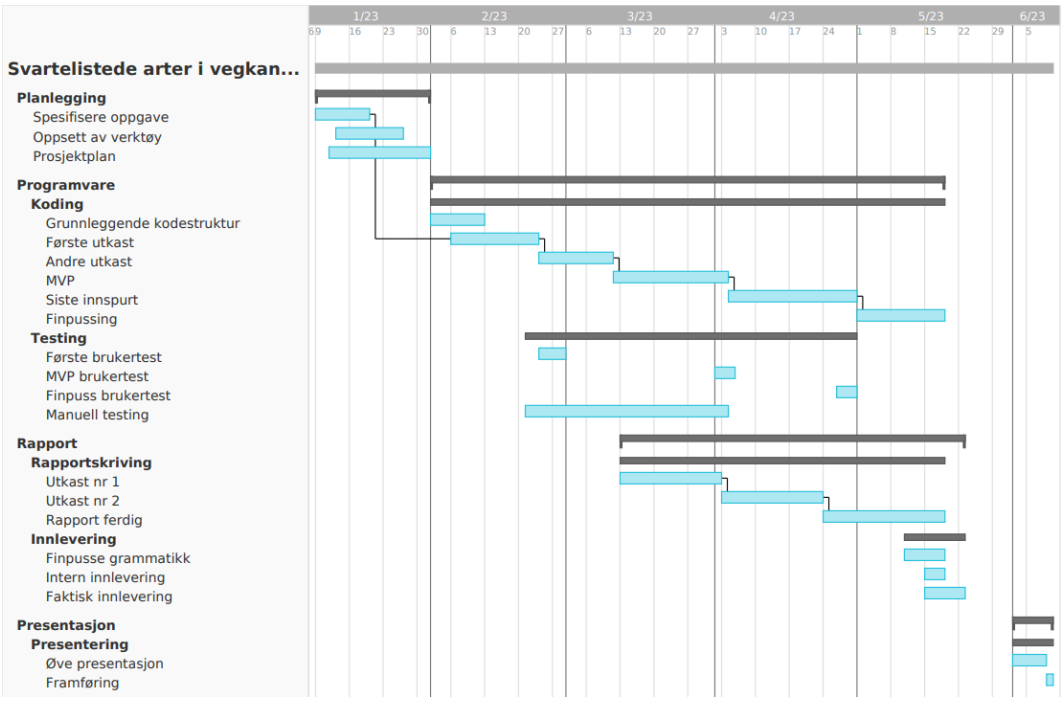
Tabell 3 viser tiltakene vi har for 6 av de 10 risikoene analysert i Tabell 2. Hver risiko blir gitt en ny sannsynlighet eller konsekvens etter tiltaket, i.e., risikoen blir analysert på nytt med tiltak.

	Tiltak	Ny vurdering av risiko etter tiltak
1	Vi vil ha møte med veileder og oppdragsgiver, for å prøve å minimalisere endringene som er gjort. Slik at arbeidet som er gjort så langt ikke trenger å endre seg veldig mye.	Usannsynlig
3	Vi kommer til gjøre regelmessige sikkerhetskopier av kildekode og rapport	Usannsynlig
4	Vi finner alternativer til teknologiene som ikke fungerer sammen. All teknologi blir sjekket på forhånd om hvorvidt den er kompatibel med det i allerede har	Usannsynlig
5	Dersom vi kommer over deler vi ikke forstår vil vi prøve å tilegne oss kunnskapen som trengs med grundig research. Den nye kunnskapen vil presenteres for gruppen slik at begge er viten i stoffet	Uproblematisk
9	Vi har skrevet opp grupperegler og rutiner vi forventer at hvert gruppemedlem følger under prosjektet	Uproblematisk
10	Vi lagrer mest sannsynlig ikke sensitive opplysninger, men hvis vi gjør det så må vi bare følge personvernlovene i forhold til hvordan vi lagrer dataene.	Uproblematisk

Tabell A.3: Tiltak på risikoer

A.6 Plan for gjennomføring

A.6.1 Gantt-skjema



Figur A.1: Gantt-skjema

A.6.2 Milepæler

- 1. februar, innlevering av prosjektplan og kontrakt
- 1. mars, første utkast og første brukertest
- 15. mars, andre utkast og andre brukertest
- 1. april, MVP ferdig
- 1. mai, hovedfunksjonalitet i programvare skal være ferdig, slik at det bare er rapport og finpussing igjen.
- 16. mai, gruppebestemt innleveringsdato, levere oppgaven, slik at vi har noen buffer dager"i tilfelle noe skulle skje de siste par dagene.
- 22. mai, faktisk innleveringsfrist
- 6.-8. juni, presentasjon

Vedlegg B

Prosjektavtale

NTNU

Norges teknisk-naturvitenskapelige universitet

Fastsatt av prorektor for utdanning 10.12.2020

STANDARDAVTALE

om utføring av studentoppgave i samarbeid med ekstern virksomhet

Avtalen er ufravikelig for studentoppgaver (heretter oppgave) ved NTNU som utføres i samarbeid med ekstern virksomhet.

Forklaring av begrep

Opphavsrett
Er den rett som den som skaper et åndsverk har til å fremstille eksemplar av åndsverket og gjøre det tilgjengelig for allmennheten. Et åndsverk kan være et litterært, vitenskapelig eller kunstnerisk verk. En studentoppgave vil være et åndsverk.

Eiendomsrett til resultater
Betyr at den som eier resultatene bestemmer over disse. Utgangspunktet er at studenten eier resultatene fra sitt studentarbeid. Studenten kan også overføre eiendomsretten til den eksterne virksomheten.

Bruksrett til resultater
Den som eier resultatene kan gi andre en rett til å bruke resultatene, f.eks. at studenten gir NTNU og den eksterne virksomheten rett til å bruke resultatene fra studentoppgaven i deres virksomhet.

Prosjektbakgrunn
Det partene i avtalen har med seg inn i prosjektet, dvs. som vedkommende eier eller har rettigheter til fra før og som brukes i det videre arbeidet med studentoppgaven. Dette kan også være materiale som tredjepersoner (som ikke er part i avtalen) har rettigheter til.

Utsatt offentliggjøring
Betyr at oppgaven ikke blir tilgjengelig for allmennheten før etter en viss tid, f.eks. før etter tre år. Da vil det kun være veileder ved NTNU, sensorene og den eksterne virksomheten som har tilgang til studentarbeidet de tre første årene etter at studentarbeidet er innlevert.

1. Avtaleparter

Norges teknisk-naturvitenskapelige universitet (NTNU) Institutt: Datateknologi og informatikk
Veileder ved NTNU: peter.nussbaum@ntnu.no 95872815
Ekstern virksomhet: Gjøvik Kommune Ekstern virksomhet sin kontaktperson, e-post og tlf.: Ingun Revhaug ingun.revhaug@gjovik.kommune.no 94803195
Student: Malene Lundemo Fødselsdato: 25.10.1999
Student: Anders Lunde Hagen Fødselsdato: 07.11.1998

Partene har ansvar for å klarere eventuelle immaterielle rettigheter som studenten, NTNU, den eksterne eller tredjeperson (som ikke er part i avtalen) har til prosjektbakgrunn før bruk i forbindelse med utførelse av oppgaven. Eierskap til prosjektbakgrunn skal fremgå av eget vedlegg til avtalen der dette kan ha betydning for utførelse av oppgaven.

2. Utførelse av oppgave

Studenten skal utføre: (sett kryss)

Masteroppgave	
Bacheloroppgave	X
Prosjektoppgave	
Annen oppgave	

Startdato: 10.01.2022
Sluttdato: 22.05.2022

Oppgavens arbeidstittel er: Svartelistede arter i vegkanten

Ansvarlig veileder ved NTNU har det overordnede faglige ansvaret for utforming og godkjenning av prosjektbeskrivelse og studentens læring.

3. Ekstern virksomhet sine plikter

Ekstern virksomhet skal stille med en kontaktperson som har nødvendig faglig kompetanse til å gi studenten tilstrekkelig veiledning i samarbeid med veileder ved NTNU. Ekstern kontaktperson fremgår i punkt 1.

Formålet med oppgaven er studentarbeid. Oppgaven utføres som ledd i studiet. Studenten skal ikke motta lønn eller lignende godtgjørelse fra den eksterne for studentarbeidet. Utgifter knyttet til gjennomføring av oppgaven skal dekkes av den eksterne. Aktuelle utgifter kan for eksempel være reiser, materialer for bygging av prototyp, innkjøp av prøver, tester på lab, kjemikalier. Studenten skal klarere dekning av utgifter med ekstern virksomhet på forhånd.

Ekstern virksomhet skal dekke følgende utgifter til utførelse av oppgaven:

Dekning av utgifter til annet enn det som er oppført her avgjøres av den eksterne underveis i arbeidet.

4. Studentens rettigheter

Studenten har opphavsrett til oppgaven[1]. Alle resultater av oppgaven, skapt av studenten alene gjennom arbeidet med oppgaven, eies av studenten med de begrensninger som følger av punkt 5, 6 og 7 nedenfor. Eiendomsretten til resultatene overføres til ekstern virksomhet hvis punkt 5 b er avkrysset eller for tilfelle som i punkt 6 (overføring ved patenterbare oppfinnelser).

I henhold til lov om opphavsrett til åndsverk beholder alltid studenten de ideelle rettigheter til eget åndsverk, dvs. retten til navngivelse og vern mot krenkende bruk.

Studenten har rett til å inngå egen avtale med NTNU om publisering av sin oppgave i NTNUs institusjonelle arkiv på Internett (NTNU Open). Studenten har også rett til å publisere oppgaven eller deler av den i andre sammenhenger dersom det ikke i denne avtalen er avtalt begrensninger i adgangen til å publisere, jf. punkt 8.

5. Den eksterne virksomheten sine rettigheter

Der oppgaven bygger på, eller videreutvikler materiale og/eller metoder (prosjektbakgrunn) som eies av den eksterne, eies prosjektbakgrunnen fortsatt av den eksterne. Hvis studenten skal utnytte resultater som inkluderer den eksterne sin prosjektbakgrunn, forutsetter dette at det er inngått egen avtale om dette mellom studenten og den eksterne virksomheten.

[1] Jf. Lov om opphavsrett til åndsverk mv. av 15.06.2018 § 1

Alternativ a) (sett kryss) Hovedregel

	Ekstern virksomhet skal ha bruksrett til resultatene av oppgaven

Dette innebærer at ekstern virksomhet skal ha rett til å benytte resultatene av oppgaven i egen virksomhet. Retten er ikke-eksklusiv.

Alternativ b) (sett kryss) Unntak

X	Ekstern virksomhet skal ha eiendomsretten til resultatene av oppgaven og studentens bidrag i ekstern virksomhet sitt prosjekt

Begrunnelse for at ekstern virksomhet har behov for å få overført eiendomsrett til resultatene: **Det er ønske om å videreføre applikasjonen etter bachelor.**

6. Godtgjøring ved patenterbare oppfinnelser

Dersom studenten i forbindelse med utførelsen av oppgaven har nådd frem til en patenterbar oppfinnelse, enten alene eller sammen med andre, kan den eksterne kreve retten til oppfinnelsen overført til seg. Dette forutsetter at utnyttelsen av oppfinnelsen faller inn under den eksterne sitt virksomhetsområde. I så fall har studenten krav på rimelig godtgjøring. Godtgjøringen skal fastsettes i samsvar med arbeidstakeroppfinnelsesloven § 7. Fristbestemmelsene i § 7 gis tilsvarende anvendelse.

7. NTNU sine rettigheter

De innleverte filer av oppgaven med vedlegg, som er nødvendig for sensur og arkivering ved NTNU, tilhører NTNU. NTNU får en vederlagsfri bruksrett til resultatene av oppgaven, inkludert vedlegg til denne, og kan benytte dette til undervisnings- og forskningsformål med de eventuelle begrensninger som fremgår i punkt 8.

8. Utsatt offentliggjøring

Hovedregelen er at studentoppgaver skal være offentlige.

Sett kryss

X	Oppgaven skal være offentlig

I særlige tilfeller kan partene bli enige om at hele eller deler av oppgaven skal være undergitt utsatt offentliggjøring i maksimalt tre år. Hvis oppgaven unntas fra offentliggjøring, vil den kun være tilgjengelig for student, ekstern virksomhet og veileder i

denne perioden. Sensurkomiteen vil ha tilgang til oppgaven i forbindelse med sensur. Student, veileder og sensorer har taushetsplikt om innhold som er unntatt offentliggjøring.

Oppgaven skal være underlagt utsatt offentliggjøring i (sett kryss hvis dette er aktuelt):

Sett kryss		Sett dato
	ett år	
	to år	
	tre år	

Behovet for utsatt offentliggjøring er begrunnet ut fra følgende:

Dersom partene, etter at oppgaven er ferdig, blir enig om at det ikke er behov for utsatt offentliggjøring, kan dette endres. I så fall skal dette avtales skriftlig.

Vedlegg til oppgaven kan unntas ut over tre år etter forespørsel fra ekstern virksomhet. NTNU (ved instituttet) og student skal godta dette hvis den eksterne har saklig grunn for å be om at et eller flere vedlegg unntas. Ekstern virksomhet må sende forespørsel før oppgaven leveres.

De delene av oppgaven som ikke er undergitt utsatt offentliggjøring, kan publiseres i NTNUs institusjonelle arkiv, jf. punkt 4, siste avsnitt. Selv om oppgaven er undergitt utsatt offentliggjøring, skal ekstern virksomhet legge til rette for at studenten kan benytte hele eller deler av oppgaven i forbindelse med jobbsøknader samt videreføring i et master- eller doktorgradsarbeid.

9. Generelt

Denne avtalen skal ha gyldighet foran andre avtaler som er eller blir opprettet mellom to av partene som er nevnt ovenfor. Dersom student og ekstern virksomhet skal inngå avtale om konfidensialitet om det som studenten får kjennskap til i eller gjennom den eksterne virksomheten, kan NTNUs standardmal for konfidensialitetsavtale benyttes.

Den eksterne sin egen konfidensialitetsavtale, eventuell konfidensialitetsavtale den eksterne har inngått i samarbeidprosjekter, kan også brukes forutsatt at den ikke inneholder punkter i motstrid med denne avtalen (om rettigheter, offentliggjøring mm). Dersom det likevel viser seg at det er motstrid, skal NTNUs standardavtale om utføring av studentoppgave gå foran. Eventuell avtale om konfidensialitet skal vedlegges denne avtalen.

Eventuell uenighet som følge av denne avtalen skal søkes løst ved forhandlinger. Hvis dette ikke fører frem, er partene enige om at tvisten avgjøres ved voldgift i henhold til norsk lov. Tvisten avgjøres av sorenskriveren ved Sør-Trøndelag tingrett eller den han/hun oppnevner.

Denne avtale er signert i fire eksemplarer hvor partene skal ha hvert sitt eksemplar. Avtalen er gyldig når den er underskrevet av NTNU v/instituttleder.

Signaturer:

Instituttleder: Dato:
Veileder ved NTNU: Dato: 25.01.
Ekstern virksomhet: Dato: 20.01.23 Jørgen Rehang
Student: Dato: 25.01.23 Anders Lunde Hagen
Student: Dato: 25.01.23 Malene Lundemo

Vedlegg C

Møtereferater

29/11/2022 - Møte med oppdragsgiver
30 min
Deltakere: Malene, Anders
første møte med oppdragsgiver. Si hei og få litt mer spesifikk info angående oppgaven.

6/1 Fredag - Møte med veileder
40 min
Deltakere: Malene, Anders, Peter
Oppstart med veileder. Møte veileder for første gang, si hei og bli kjent. Etablere ukentlige møter og diverse forventinger til bacheloroppgaven

09/01 Mandag - Sprint planning
4 timer
Deltakere: Malene, Anders
Første sprint planning møte. Starte smått med oppretting av teams rom, git repo og tittet litt på kontraktene vi skal skrive.

11/01 Onsdag- Møte med veileder
30 min
Deltakere: Peter, Malene, Anders
Møte med veileder Snakket om starten på oppgaven. Hvordan vi burde gå fram med starten av oppgaven. Tips til hvordan vi skal starte oppgaven og ting som vi burde fokusere på i starten. Litt om hva vi burde snakke med Ingun om videre.

12/01 Torsdag
3 timer
Deltakere: Anders, Malene
Startet på prosjektplanen. Ikke mye oppg info så langt, så fokuserte på punkter som grupperegler, Gantt og introduksjon, som ikke omhandlet selve oppgaven i detalj.

13/01 Fredag - Sprint review med oppdragsgiver + retrospektivt
1 time
Deltakere: Anders, Malene, Ingun
Oppstartsmøte med oppgavegiver. Diskuterte oppgaven, kontrakten (standardavtalen), samt ressurser vi kunne benytte i kommunen. Diskuterte hvordan sprintene fungerer, fungerer bra så langt.

16/01 Mandag - Sprint planning
4 timer
Deltakere: Malene, Anders
Fortsettet arbeidet med prosjektplanen. Planlagt hvikle punkter som er viktigst å fylle ut på prosjektplan.

18/01/2022 Onsdag - Møte med veileder

1 time
Deltakere: Anders, Malene, Peter
Fikk tips om hvordan snakke/planlegge med oppdragsgiver. Satte frist for rapportutkast

20/01 Fredag - Møte med oppdragsgiver
3 timer
Deltakere: Malene, Anders, Ingun, +kommunale ansatte
Møte med oppdragsgiver og 2 stk fra kommunen. Snakket om dagens løsning og hvordan en eventuell ny digital løsning kan bli seende ut. Hva som er ønsket, hva som er best, og hva som ikke nødvendigvis trengs, i forhold til både løsningen for Innbyggere, men også traktorførere og ansatte som mottar henvendelser

20/01 Fredag - Sprint review
30 min
Deltakere: Anders og Malene
Diskuterte møte med oppdragsgiver. Oppsummerte de nye punktene/endringene i løsningen som er tenkt.

23/01 Mandag - Sprint planning
2 timer
Deltakere: Anders og Malene
Fylt inn siste rest av prosjektplan før første utkast leveres til veileder. Fortsatt et par punkter som ikke er fylt inn, da vi venter på møte med IKT ansatt i kommunen for hvordan vi skal løse det tekniske med nettsider og lignende. Planlagtto møter med noen fra kommunen sammen med oppdragsgiver.

24/01 Tirsdag - Møte med oppdragsgiver
1 time
Deltakerne: Anders, Malene, Ingun, Alexander
Møte med en fra kommunen som hjalp oss i forhold til oppretting av skjema og eksisterende løsninger som finnes i dag som vi eventuelt kan bruke.

25/01 Onsdag - Møte med veileder
30 min
Deltakere: Anders, Malene, Peter
Gikk igjennom prosjektrapport og veilder kom med forslag til endringer. Skrev under på prosjektavtalen.

25/01 Onsdag
30 min
Deltakere: Anders, Malene
Scannet inn dokumenter og fordelte prosjektrapport-skrivingen hvor endringer skal gjøres før innlevering

27/01 Fredag - Møte med oppdragsgiver

1,5 timer
Deltakere: Anders, Malene, Ingun, Bård, Karl Andreas
Møte angående potensielle kartløsninger. Så på nåværende løsninger og om vi kunne hive oss på dem. Det er godt potensial, men må ha et møte til før vi får etablert noe fast

27/01 Fredag - Sprint review + retrospektivt
30 min
Deltaktere: Malene, Anders
Hatt to møter med kommunen. Fått mye god info som vi må bearbeide. Ukentlige sprinter fungerer veldig bra, og vil flytte retrospektive møter til hver 3. uke istedenfor hver 2. uke som var den originale planen.

30/01 Mandag - Møte med kommunen uten oppdragsgiver
1 time
Deltakere: Anders, Malene, Pål Godard, Bård
Diskuterte mulige avgrensninger og rammer for vårt prosjekt. Ble enig om å tenke mer kommersielt og åpent, i.e., ikke lukke oss til Gjøvik kommunes systemer men lage en API løsning av noe slag som kan brukes av flere kommuner.

30/01 Mandag - Sprint planning
30 min
Deltakere: Anders, Malene
Nærmer oss slutten av planleggingsperioden, må gjøre ferdig prosjektplanen. Dermed begynne på oppsett av nettsidene.

31/01 Tirsdag
1 time
Deltakere: Anders, Malene
Fullførte prosjektplan og leverte til veileder.

01/02 Onsdag - Møte med veileder
30 min
Deltakere: Anders, Malene, Peter
Så på prosjektplan, den er godkjent, men vi fikk forslag til noen ekstra endringer vi kan gjøre for å gjøre prosjektplanen fin til den endelige rapporten som skal leveres i mai.

02/03 - Torsdag
30 min
Deltakere: Malene, Anders
Lite ekstra møte for å planlegge hva vi kan gjøre i dag og i morgen med tanke på oppsett av arbeidsmiljøet vårt.

03/02 Fredag - Sprint review
30 min

Deltakere: Malene, Anders
Undervurderte litt hvor lite vi hadde å gjøre etter prosjektplanen var ferdig. Hadde et lite ekstra møte på torsdagen for å se hva vi kan gjøre før neste uke.

06/02 Mandag - Sprint planning
1 time
Deltakere: Anders, Malene
Planla uken, satt opp git-repo og server. Begynte å se på hva vi trenger til uken for å sette opp et fungerende arbeidsmiljø mellom oss.

09/02 Torsdag - Møte med veileder
30 min
Deltakere: Anders, Malene, Peter
Diskuterte hva vi har kommet frem til at vi skal lage med veileder, samt hva vi har tenkt å gjøre fremover. Fikk litt tips om hvordan vi kan gå fram med løsningen, e.g., starte med å få opp en liten framework vi kan bygge på slik at vi ikke starter med alt samtidig, det er bedre å få noe gjort ferdig enn ingenting ferdig

10/02 Fredag - Sprint review
30 min
Deltakere: Malene, Anders
Fikk gjort mye denne uken, satt opp mye med tanke på videre arbeid. Fikk gjort alt vi planla, og var et godt estimat av hvor mye arbeid som skulle bli gjort denne uken.

13/02 Mandag - Sprint planning
1 time
Deltakere: Malene, Anders
Startet ny sprintuke. Begynt mer på oppsett av nettsidene. Satt opp flere issues til uken som kommer, med tanke på at vi ønsker å vise fram nettsiden i morgen

14/02 Tirsdag - Møte med oppdragsgiver
1 time
Deltakere: Ingun, Malene, Anders
Oppdateringsmøte med oppgavegiver. Har ikke hatt møte på 2 uker, og vi hadde et møte med noen andre fra kommunen uten oppgavegiver, så vi hadde et lite møte hvor vi oppdaterte henne og viste fram hvordan vi ligger an så langt. Fikk litt input på hva vi burde jobbe med videre.

15/02 Onsdag - Møte med veileder
1 time
Deltakere: Peter, Anders, Malene
Møte med veileder, bare snakket om hvordan vi ligger an og hva planene er de neste ukene. Viste hvor langt vi har kommet og snakket litt om ideer på hva vi burde gjøre videre fremover

17/02 Fredag - Sprint reveiew + retrospektivt
30 min
Deltakere: Anders, Malene
Har begynt å få en god ide om hvor mye arbeid vi burde planlegge hver uke. Siste par ukene har vært bra, med god mengde arbeid. Ukentlige sprinter var et godt valg. Denne uken fikk vi mye gjort med tanke på å få en nettside vi kan vise fram.

20/02 Mandag - Sprint planning
30 min
Deltakere: Anders, Malene
Planlagt videre arbeid med nettsidene. Se litt mer på adminsidene og vurdere brukerskjema design.

22/02 Onsdag - Møte med veileder
30 min
Deltakere: Peter, Anders, Malene
Viste fram litt av det vi hadde, diskuterte hva vi kunne legge inn i rapporten av fremtidige ideer vi kanskje ikke rekker å legge til i prosjektet

24/02 Fredag - Sprint review
30 min
Deltakere: Anders, Malene
Behandlet merge requests og merge conflicts. Hadde et problem med merginen av to brancher, som tok lengre tid enn vi trodde. Men fikk løst det og gikk greit til slutt. Ellers har uken vært effektiv. Vi så også på eventuelle løsninger som ikke er mulig for oss med så kort prosjekttid, men noterer det ned til rapporten.

27/02 Mandag - Sprint planning
1 time
Deltakere: Anders, Malene
Startet ny sprint. Vi planla uken og fordelte oppgaver, samt diskuterte hva vi fikk til forrige uke. Snakket om litt designendringer og lignende på nettsidene og at vi må begynne med brukertesting da vi begynner å ha et fungerende produkt.

01/03 Onsdag - Møte med veileder
30 min
Deltakere: Peter, Anders, Malene
Viste nye resultat og diskuterte de med veilder. Hadde noen gode ideer om hva vi kan jobbe med videre.

02/03 Torsdag - Møte med oppdragsgiver
1 time
Deltakere: Anders, Malene, Peter, Ingun
Demo av prosjektet så langt. Diskuterte design-valg og elementer oppdragsgiver

ønsker og ikke ønkser i GUI-en

03/03 Fredag - Sprint review
30 min
Deltakere: Anders, Malene
Fikk testet litt denne uken som var positivt. Viste også fram til oppdragsgiver og veileder og begge to kom med gode innspill til videre arbeid. Ellers er vi på god vei i forhold til hva oppgaven ber om.

06/03 Mandag - Sprint planning
1 time
Deltakere: Anders, Malene
La en plan for uken om at vi kan begynne å titte på rapporten med tanke på at den må begynnes på en eller annen gang. Ellers fortsette med arbeid på nettsidene. Har også tenkt å se på database alternativer til uken.

08/03 Onsdag - Møte med veileder
1 time
Deltakere: Anders, Peter
Møte med veileder hvor vi snakket om begynnelse på rapport og status så langt. Litt idemyldring i forhold til hva vi burde prioritere videre.

10/03 Fredag - Møte med oppdragsgiver
2 timer
Deltakere: Anders, Malene, Ingun, og to andre fra kommunens landbruksavdeling
Viste demo 2. Diskuterte design og ønsker på kart. Snakket om logoer og bruk av farger i kartet. Kom med mange forslag om hva de i kommunen trenger og har lyst på og hva det viktigste er for dem å få av informasjon fra brukeren. Mye viktig info som vi ikke hadde fra før, i tillegg til innspill fra andre enn oppdragsgiveren som var positivt med tanke på at dette skal være et produkt for flere personer.

10/03 Fredag - Sprint review + retrospektivt
30 min
Deltakere: Anders, Malene
En god uke, da vi fikk masse innspill på design og funksjonalitet på nettsidene. Ellers har ukene gått bra og ukentlige sprinter fungerer fortsatt veldig bra. Retrospektivt hver 3. uke fungerer også bra og vil beholde det fremover.

13/03 Mandag - Sprint planning
30 min
Deltakere: Anders, Malene
Planla litt med tanke på endringeen vi snakket om forrige uke. Fått litt mer innblikk i hva sluttproduktet kan være, så begynt å se mer på testing og hva som skjer når man når et sluttprodukt.

16/03 Torsdag - Møte med veileder
2 timer
Deltakere: Anders, Malene, Peter
Møte med veileder, snakket om ting vi burde skrive i rapporten som testing og hva vi ville gjort i en eventuell deployment av koden til Gjøvik, hvordan reklamerer vi for koden?

17/03 Fredag - Sprint review
30 min
Deltakere: Anders, Malene
Fin uke, fikk lagt til mye av det vi planlagte. Sett litt på arbeidsmengden og den ligger bra an.

20/03 Mandag - Sprint planning
30 min
Deltakere: Anders, Malene
Til uken vil vi se mer på videre arbeid innenfor adminsidene. Skal skrive ned mer i forhold til rapporten, og legge ved notater på design endringer og lignende for enklere rapportskriving

22/03 Onsdag - Møte med veileder
2 timer
Deltakere: Anders, Malene, Peter
Møte med veileder, snakket om potensialet til AI og hvordan det kunne teoretisk blitt brukt i oppgaven vår.

24/03 Fredag - Sprint review
30 min
Deltakere: Anders, Malene
Snakket en del om sluttproduktet og videre arbeid, og hvordan vi burde legge til det i rapporten. Ellers ligger nettsidene godt an, har fått en del på plass nå og skal planlegge møte neste uke med oppdragsgiver

27/03 Mandag - Sprint planning
1 time
Deltakere: Anders, Malene
Sprintmøte, planlagt uken. Snakket om kommende uker i forhold til at vi nærmer oss innlevering. Planla et møte med veileder denne uken slik at vi kan vise at vi nærmer oss slutten og hva de viktigste aspektene av oppgaven er igjen for oss å gjøre ferdige

29/03 - Onsdag
1 time
Deltakere: Malene, Peter
Møte med veileder. Diskuterte nye endringer i programmet, samt database

oppsett. Snakket om viktige punkter å ha med i rapporten; modeller, designvalg, diskusjoner vi har hatt, valg vi har tatt og hvorfor vi endte opp med det vi endte opp med.

31/03 Fredag - Møte med oppdragsgiver
1 time
Deltakere: Malene, Anders, Ingun
4. demo visning. Diskuterte oppretting av PDF med info om planter som skal fjernes, aka. hva skjer når bruker trykker "till bekjemping"

31/03 Fredag - Sprint review + retrospektivt
30 min
Deltakere: Anders, Malene
Veldig effektiv uke. I tillegg til å nærme oss ferdig med programmet, har vi også begynt smått på rapporten og legge ved ting vi burde skrive om. Sprintmøtene har også fungert bra så langt. Gode arbeidsmengder og jevnlige møter fungerer bra.

03/04 Mandag - Sprint planning
30 min
Deltakere: Anders, Malene
Påsken er rett rundt hjørnet, så vi har sett på det viktigste vi må gjøre ferdig før påsken. Ingen veiledning denne uken.

07/04 (lang)Fredag - Sprint review
10 min
Deltakere: Anders, Malene
En kort uke pga påske, men tok et kjapt møte siden vi jobbet et par av dagene. En kort uke, men fikk gjort unna et par viktige issues som er kritiske for sluttproduktet.

11/04 Tirsdag - Sprint planning
30 min
Deltakere: Anders, Malene
Pga 2. påskedag har vi sprint planning i dag, vi nærmer oss slutten med stormskritt så vi har begynt å planlegge prioriteringer vi må gjøre. Vi skal begynne å skrive rapporten snart, så nettsidene må nærme seg ferdig

12/04 Onsdag - Møte med veileder
1 time
Deltakere: Malene, Anders, Peter
Snakket om sluttperioden av bacheloroppgaven. Vi nærmer oss ferdig med programvare og begynner sikkert å gå over til rapport og vekk ifra nye implementeringer på nettsiden neste uke

14/04 Fredag - Sprint reveiw

30 min
Deltakere: Anders, Malene
En god uke, vi nærmer oss et sluttprodukt som virkelig tar en god form. Vi har jobbet solid med issues og har hatt en god uke siden etter påsken. Har bare et par punkter igjen før nettsidene kan i teorien deployes som et sluttprodukt som fungerer for kommunen.

17/04 Mandag - Sprint planning
1 time
Deltakere: Anders, Malene
Sprintmøte om kommende uke. Nettsidene skal visuelt se ferdige ut for oppdragsgiver, da vi sier oss ferdige for oppdragsgiver sin del denne uken. Begynne å fokusere på å skrive rapporten, i tillegg til små endringer i koden som ikke angår oppdragsgiver. Typ kommentarer, små bug fiksing, navngiving etc...

19/04 Onsdag - Møte med oppdragsgiver
1 time
Deltakere: Malene, Anders, Ingun
Viste fram siste endringene på nettsiden. Snakket om at rapporten er fokus og lite endringer blir gjort fremover. Snakket om muligheter i fremtiden om programmet faktisk skal tas i bruk. Oppdragsgiver er fornøyd og snakket om hvordan nettsidene løser de problemene hun har den dag i dag. Var også ønske om at dette faktisk blir gitt til kommunen slik at hun kan bruke løsningen. Uansett om det skjer eller ikke, er det en veldig hyggelig beskjed å få mot slutten av prosjektet :)

19/04 Onsdag - Møte med veileder
1 time
Deltakere: Malene, Anders, Peter
Snakket om første utkast rapport. Fikk litt tilbakemelding og snakket om hva vi burde skrive på videre. Viste siste endringene gjort på nettsiden og snakket om samtalen vi hadde med oppdragsgiver.

21/04 Fredag - Sprint review + retrospektivt
30 min
Deltakere: Anders, Malene
En god uke, har gjort ferdig nettsidene og har fått en god start på rapporten. De siste par ukene har også gått veldig bra, har hatt god arbeidsflyt og lagt opp til fine arbeidsmengder hver uke. Nærmer oss slutten og har lite vi skulle ønske vi forandret på i etterkant.

24/04 Mandag - Sprint planning
30 min
Deltakere: Anders, Malene
Hovedfokus denne uken er rapport. Litt smått kode og database, men i all hovedsak få skrevet en del i rapporten.

26/04 Onsdag - Møte med veileder
30 min
Deltakere: Anders, Malene, Peter
Gått igjennom utakst av rapport. Veileder hjelper med hvilke punkter vi burde fokusere på og hvikle som ikke er så viktige med tanke på tidsbruk.

28/04 Fredag - Sprint review
30 min
Deltakere: Anders, Malene
Fått skrevet en del på rapporten, ligger godt an i forhold til innleveringsfrist. Arbeidsmengden var god og vil sikte på samme mengde fremover.

01/05 Mandag - Sprint planning
30 min
Deltakere: Anders, Malene
Vil hovedsaklig jobbe med rapport. Vi fant en litt kritisk bug i helgen, så vil også bruke tid på å fikse det i koden.

03/05 Onsdag - Møte med veileder
1 time
Deltakere: Malene, Anders, Peter
Snakket om videre arbeid i rapport. Fikk tilbakemelding på utkast.

05/05 Fredag - Sprint review
30 min
Deltakere: Anders, Malene
Til tross for å ha funnet en bug, fordelte vi arbeidsmengden bra mellom buggen og rapporten. Buggen tok ikke så lang tid å fikse, så den tok ikke over mye av rapportskrivingen, men begge måtte hjelpe for å fisks buggen så ikke ideelt.

08/05 Mandag - Sprint planning
30 min
Deltakere: Anders, Malene
Rapporten nærmer seg ferdig. Har bare litt på noen punkter igjen og avslutningen. Denne uken skal vi gjøre ferdig alle, utenom avslutning, men vil fylle inn litt på avslutning mot slutten av uken.

10/05 Onsdag - Møte med veileder
1 time
Deltakere: Malene, Anders, Peter
Nærmer seg slutten. Tilbakemelding på nest siste utkast. Nesten alt er skrevet unnatt konklusjon og småting som vedlegg og gloser.

12/05 Fredag - Sprint review + retrospektivt
30 min

Deltakere: Anders, Malene
God uke. Har skrevet ferdig på alt, unntatt avslutning. Vil skrives ferdig neste uke. Siste retrospektive møtet og er veldig fornøyd med Scrum oppsettet vårt. Dette er også siste referat på rapport, da referatene vil bli formatert og lastet opp i rapporten til helgen.

Vedlegg D

Timelogg

Uke 2	Malene	Anders
Planlegging	15	14
Research	7	7
Dokumentasjon	12	11
Koding		
Testing		
Sum	34	32
Kum	34	32

Uke 3	Malene	Anders
Planlegging	8	9
Research	7	7
Dokumentasjon	16	13
Koding		
Testing		
Sum	31	29
Kum	65	61

Uke 4	Malene	Anders
Planlegging	9	6
Research	3	6
Dokumentasjon	22	19
Koding		
Testing		
Sum	34	31
Kum	99	92

Uke 5	Malene	Anders
Planlegging	4	3
Research	1	2
Dokumentasjon	3	4
Koding	25	24
Testing		
Sum	33	33
Kum	132	125

Uke 6	Malene	Anders
Planlegging	4	1
Research	2	1
Dokumentasjon	0	2
Koding	24	28
Testing		
Sum	30	32
Kum	162	157

Uke 7	Malene	Anders
Planlegging	0	1
Research	1	0
Dokumentasjon	5	3
Koding	24	29

Testing		
Sum	30	33
Kum	192	190

Uke 8	Malene	Anders
Planlegging	4	4
Research	5	5
Dokumentasjon	4	4
Koding	18	19
Testing		
Sum	31	32
Kum	223	222

Uke 9	Malene	Anders
Planlegging	1	2
Research	4	4
Dokumentasjon	4	3
Koding	22	18
Testing	1	3
Sum	32	30
Kum	255	252

Uke 10	Malene	Anders
Planlegging	0	0
Research	1	5
Dokumentasjon	4	4
Koding	24	22
Testing	2	2
Sum	31	33
Kum	286	285

Uke 11	Malene	Anders
Planlegging		
Research	1	1
Dokumentasjon	4	3
Koding	22	21
Testing	5	4
Sum	32	29
Kum	318	314

Uke 12	Malene	Anders
Planlegging		
Research		1
Dokumentasjon	2	2
Koding	27	25
Testing	2	1
Sum	31	29
Kum	349	343

Uke 13	Malene	Anders

Planlegging		
Research	2	
Dokumentasjon	3	2
Koding	25	29
Testing	2	
Sum	32	31
Kum	381	374

Uke 14	Malene	Anders
Planlegging		
Research	3	2
Dokumentasjon	3	3
Koding	12	13
Testing	4	3
Sum	22	21
Kum	403	395

Uke 15	Malene	Anders
Planlegging		
Research	6	6
Dokumentasjon	5	5
Koding	19	18
Testing	2	2
Sum	32	31
Kum	435	426

Uke 16	Malene	Anders
Planlegging		
Research	2	2
Dokumentasjon	13	14
Koding	8	12
Testing	3	4
Sum	26	32
Kum	461	458

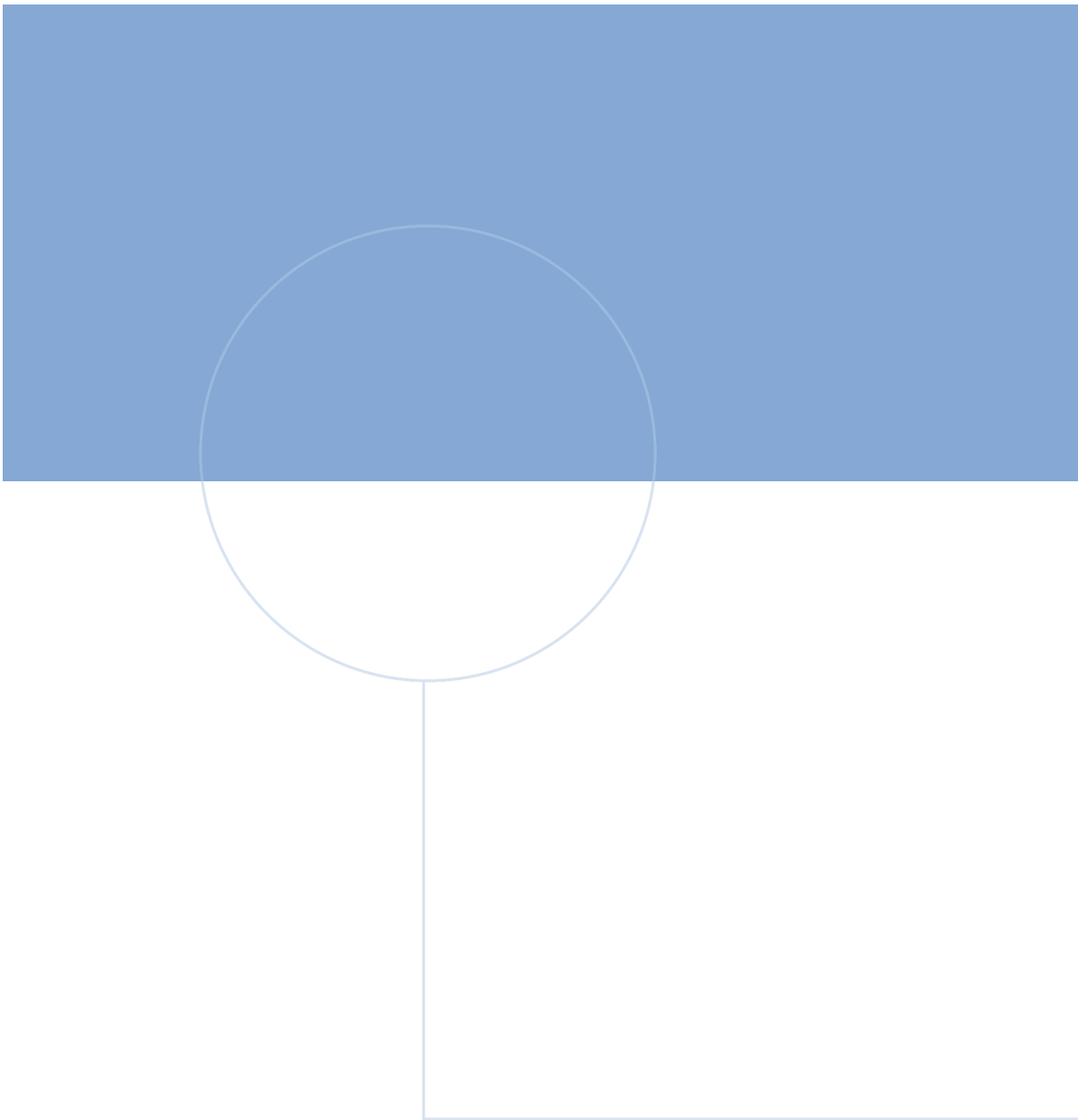
Uke 17	Malene	Anders
Planlegging		
Research	4	4
Dokumentasjon	19	19
Koding	7	6
Testing	1	1
Sum	31	30
Kum	492	488

Uke 18	Malene	Anders
Planlegging		
Research	5	6
Dokumentasjon	23	25
Koding	1	1
Testing		

Sum	29	32
Kum	521	520

Uke 19	Malene	Anders
Planlegging		
Research	4	3
Dokumentasjon	28	27
Koding		
Testing		
Sum	32	30
Kum	553	550

Uke 20	Malene	Anders
Planlegging		
Research	2	2
Dokumentasjon	29	30
Koding		
Testing		
Sum	31	32
Kum	584	582



NTNU

Kunnskap for en bedre verden