Bacheloroppgave

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

Pedersen, Markus Solli
Roness, Nicolay Caspersen
Taklo, Leonard Opsal

Digitalisering av NTNU TBM Modellen

Bacheloroppgave i Ingeniørfag, data
Veileder: Norozi, Muhammad Ali
Mai 2023

NTNU
Kunnskap for en bedre verden

Pedersen, Markus Solli
Roness, Nicolay Caspersen
Taklo, Leonard Opsal

Digitalisering av NTNU TBM Modellen

Bacheloroppgave i Ingeniørfag, data
Veileder: Norozi, Muhammad Ali
Mai 2023

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

NTNU
Kunnskap for en bedre verden

Sammendrag

Denne bacheloroppgaven handler om å digitialisere doktoravhandlingen 'Hard Rock Tunnel Boring: Performance Predictions and Cutter Life Assessment' av Francisco Javier Macias [1]. NTNU-modellen for TBM (Tunnel Boring Machine) er en matematisk modell for TBM-prosjekter i harde bergarter. Modellen tar utgangspunkt i empirisk data fra tunnelprosjekter og beskriver hvordan forskjellige maskin- og geologi-parametere har en effekt på maskinens fremdrift og levetiden på kutterene. Fra før av finnes det programvare for modellen, men denne programvaren er enten utdatert, mangler ønsket funksjonalitet eller gjør feil i beregninger. Å digitalisere modellen har som mål å bespare tid i utregninger, gi mer pålitelige resultater, gjøre det enklere å samarbeide i TBM-prosjekter og bruke modellen i undervisningsammenheng.

Gjennom bruk av den iterative utviklingsmetodikken Scrum og utviklingsmetoder som wireframing og prototyping har gruppen utviklet en webapplikasjon med blant annet Vue, Typescript, Java, Spring Boot og MySQL. Applikasjonen som er utviklet støtter et brukersystem, prosjektsystem, e-post, norsk og engelsk språkstøtte og administrering av parametere, språk og brukere. Implementasjonen av NTNU-modellen står som den sentrale delen av systemet og består av 81 parametere med 49 tilhørende matematiske funksjoner hvor en endring av et parameter forplanter seg gjennom systemet og viser den oppdaterte modellen. For å bestå kravet om å være en sikker og brukervennlig applikasjon er den evaluert med WCAG og OWASP Top 10, og for å sikre at modellen gir pålitelig resultater har de matematiske funksjonene en testdekning på 84%.

Systemet som er utviklet i tett samarbeid med produkteier kan anses som komplett og pålitelig både i opplæringssammenheng og for bransjen generelt. Denne rapporten beskriver relevant teori for å forstå problemstillingen og resultatet av denne. Til tross for et system som stort sett består kravene i henhold til visjonsdokumentet gjenstår det fortsatt videre arbeid med applikasjonen som implementasjon av usikkerhet og kostnadsberegninger.

Abstract

This bachelor thesis focuses on digitizing the doctoral dissertation 'Hard Rock Tunnel Boring: Performance Predictions and Cutter Life Assessment' by Francisco Javier Macias [1]. The NTNU-model for TBM (Tunnel Boring Machine) is a mathematical model for TBM-projects in hard rock formations. The model is based on empirical data from tunnel projects and describes how various machine and geological parameters affect the machine's progress and cutter life. Existing software for the model is either outdated, lacks desired functionality, or contains calculation errors. The goal of digitalizing the model is to save time in calculations, provide more reliable results, facilitate collaboration in TBM projects, and to use the model for educational purposes.

Using the iterative development methodology Scrum and development techniques such as wire-framing and prototyping, the group has developed a web application using technologies like Vue, Typescript, Java, Spring Boot, and MySQL. The developed application supports user management, project management, email notifications, Norwegian and English language support, and administration of parameters, languages, and users. The implementation of the NTNU model is the central part of the system, consisting of 81 parameters with 49 associated mathematical functions. Any change in a parameter propagates through the system, reflecting the updated model. To meet the requirements of being a secure and user-friendly application, the applications has been evaluated with WCAG and OWASP Top 10. In order to ensure reliable results from the model, the mathematical functions has a test coverage of 84%.

The system, developed in close collaboration with the product owner, can be considered complete and reliable, both for educational purposes and for the industry in general. This report describes the relevant theory to understand the problem statement and presents the results. Despite the system largely meeting the requirements outlined in the vision document, further work still remains, particularly regarding the implementation of uncertainty and cost calculations.

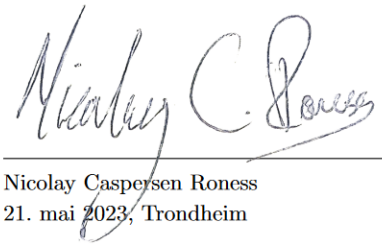
Forord

Bacheloroppgaven vi har valgt er NTNU TMB-videreutvikling, som baserer seg på doktoravhandlingen 'Hard Rock Tunnel Boring: Performance Predictions and Cutter Life Assessment' av Francisco Javier Macias [1]. Oppgaven ble valgt på grunnlag av en grundig elimineringsprosess hvor flere faktorer ble tatt høyde for. Den første faktoren var gjennomførbarhet. Om vi så det som realistisk å kunne ferdigstille oppgaven som vi valgte. En annen viktig faktor for oss var at oppgaven var tverrfaglig. Vi mente at vi kunne oppnå viktig kunnskap om vi fikk muligheten til å jobbe med fagfolk innen andre felt, samt sette oss inn i teori og metoder fra andre fagfelt. Til slutt var det også viktig for oss å jobbe på med en oppgave som var til stor nytte for industrien og som hadde potensiale til å bli brukt daglig av andre mennesker.

Resultatet har blitt oppnådd gjennom en prosess bestående av litteratursøk av NTNU TBM-modellen, iterativ utviklingsmetode, tett kommunikasjon og oppfølging av produkteiere og valg og bruk av passende teknologi i henhold til problemstillingen.

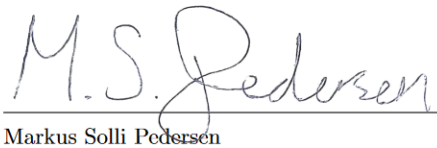
Vi ønsker å takke våre oppgavestillere Amund Bruland og Helge-Ivar Frostad for et godt samarbeid og tett oppfølging. Bruland har gjennom hele prosessen veildet gruppen innen TBM-faget, og har vært en viktig beslutningstaker i designvalg av systemet. Frostad sitt arbeid med regresjoner og videreutvikling av grafene vist i NTNU-modellen har vært kritisk for digitaliseringsprosessen.

Til slutt vil vi også takke vår veileder Muhammad Ali Norozi for god veiledning innenfor systemutvikling gjennom våren, og for å være en god støttespiller under dokumentasjonsarbeid og skriving av hovedrapport.



Nicolay Caspersen Roness
21. mai 2023, Trondheim

Leonard Opsal Taklo
21. mai 2023, Trondheim



Markus Solli Pedersen
21. mai 2023, Trondheim

Oppgavetekst

Original oppgavetekst fra Vedlegg J:
'Videreutvikle et nettbasert verktøy for prediksjon av produksjon og kostnad ved fullprofilboring av tunneler [...] Vi ønsker oss et brukervennlig og nettbasert verktøy som baseres på Javier Macias sin phd-avhandling. Verktøyet som ble utviklet i 2017 har noe av den ønskede funksjonaliteten, men det gjenstår mye på analyse, risiko med mer. Også integrering av andre beregningsmodeller enn NTNU sin. At verktøyet legger til rette for nettbasert samhandling, er særlig viktig i tidligfasen av tunnelprosjekter, siden TBM-teknologien og -bransjen er internasjonal. Det er ønskelig at applikasjonen kan simulere usikkerhet i beregningsmetode (ulike beregningsmodeller) og i inngangsparametere.'

Basert på oppgaveteksten ble det definert en problemstilling og funksjonelle krav i Vedlegg C: Visjonsdokument og Vedlegg D: Kravdokumentasjon. Kostnadsutregning ble fjernet fra kravlisten underveis i prosjektets gang grunnet prioritering av NTNU-modellen.

Innholdsfortegnelse

Figurer

Tabeller

1 Introduksjon Og Relevans

Introduksjon og relevans omhandler oppgavens relevans, problemstillingen, struktur på rapporten og akronymer og forkortelser.

1.1 Relevans

Tunnelboremaskiner brukes til tunnelkonstruksjon som et alternativ til drilling og sprenging, og innebærer en større initiell kostnad på grunn av innkjøpet av maskinen. Det er derfor viktig i planleggingsfasen av et tunnelprosjekt å ha et godt verktøy som kan svare på om den store initielle investeringen er verdt det. Det finnes flere slike verktøy som modellerer et TBM-prosjekt, der en av de ledende modellene kommer fra NTNU. NTNU-modellen for TBM [1] beskriver forhold mellom maskin- og geologi-parametere og deres effekt på fremdriften og kutterdiskenes levetid. Den er derimot et dokument med matematiske formler og grafer, så det innebærer mye manuell regning å bruke modellen direkte. Det er derfor et behov for en digital versjon av NTNU-modellen for TBM.

Det finnes programvare basert på NTNU-modellen fra før, men disse er enten utdatert, mangler ønsket funksjonalitet eller gjør feil i beregninger. Ved å produsere en ny, fungerende og oppdatert digitalisering av NTNU-modellen vil dette verktøyet bespare tid ved beregninger, gi pålitelige resultater, gjøre det enklere å samarbeide i TBM-prosjekter, og fungere som et verktøy til undervisning i faget.

1.2 Problemstilling

Problemstilling for bacheloroppgaven er:

Digitalisering av NTNU TBM-modellen for bruk i prosjektplanlegging og undervisning.

Videre har problemstillingen blitt delt inn i delmålene i henhold til Vedlegg C: Visjonsdokument:

- NTNU-Modellen for TBM
- Brukersystem
- Prosjektsystem
- Administratorpanel
- Språkstøtte
- Epost

1.3 Struktur

Kapittel 1 - Introduksjon og relevans Introduksjon til problemstillingen, relevansen til oppgaven, struktur til rapporten og akronymer og forkortelser.

Kapittel 2 - Teori og relevant litteratur Presenterer teori og relevant litteratur som er viktig for å forstå resten av rapporten.

Kapittel 3 - Metode Presenterer forskningsmetode og utviklingsmetode som blir brukt for å komme frem til resultatet.

Kapittel 4 - Resultater Presenterer resultatene i henhold til kravene i Vedlegg C: Visjonsdokument.

Kapittel 5 - Diskusjon Tar for seg og drøfter hvorfor resultatene ble som det ble.

Kapittel 6 - Konklusjon og videre arbeid Konklusjon av rapporten og hva som kan gjøres av videre arbeid.

Samfunnspåvirkning Drøfter arbeidet som har blitt gjort i forhold til et helhetlig systemperspektiv.

1.4 Akronymer Og Forkortelser

- TBM Tunnelboremaskin
- CSR Controller-Service-Repository
- CRUD Create, Read, Update, Delete
- JPA Java Persistence API
- API Application Programming Interface
- JWT JSON Web Token
- CLI Cutter Life Index
- DRI Drilling Rate Index
- OOFS Orientation Of Fracture Set
- PR Pull Request
- LCA Life Cycle Assessment

2 Teori Og Relevant Litteratur

Teori og relevant litteratur omhandler teori som er viktig for å forstå de andre delene av rapporten. Viktig teori angående NTNU-Modellen for TBM, arkitekturmønster CSR, livsløpsvurdering og tidligere programvare for NTNU-Modellen blir beskrevet.

2.1 NTNU-Modellen For TBM

2.1.1 Tunnelboremaskiner

En tunnelboremaskin (TBM) er en sylinderformet maskin som freser ut steinmasse for boring av tunneller. Maskinen graver seg gjennom fjellet og etterlater en tunnel med diameter lik diameteren til borhodet. I front av TBM'en er et roterende borhode som trykkes mot fjellveggen ved hjelp av hydraulikk. På borhodet er det flere skiver, eller disker, som skraper og knuser steinmassen [2]. Kutterdiskene blir slitt etterhvert som de freser ut steinmasse, og må regelmessig byttes ut. Det fins flere typer maskiner for ulike behov, maskinene NTNU-modellen beskriver er av typen egnet mot harde bergarter, kalt 'hard rock TBM' eller 'main beam TBM' på engelsk, og er 3 til 12 meter i diameter.

Boreprossessen består av at maskinen griper seg fast i sidene av tunnelen, skyver borhodet fremover mot fjellet, og roterer borhodet slik at kutterdiskene freser steinmasse fra fjellet. Når maskinen har utvidet borhodet til sitt maksimum slipper den grepet fra sidene av tunnelen og beveger seg fremover mens borhodet trekkes tilbake, slik at den er klar for å skyve borhodet frem igjen [3].

Figur 1: En TBM på 17.4m brukt i Washington [4]

2.1.2 NTNU-Modellen

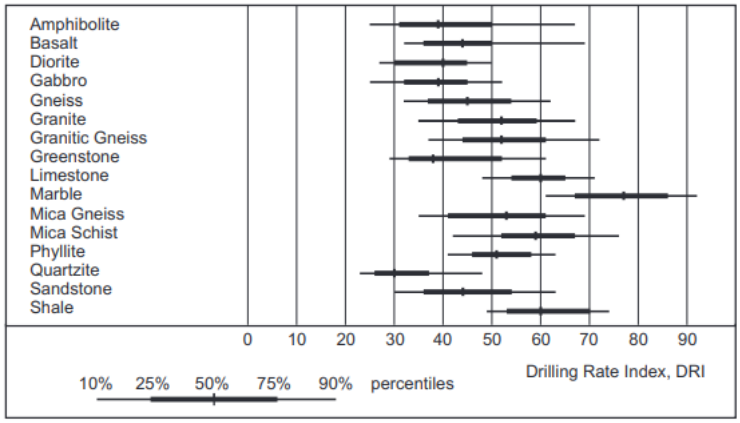
NTNU-modellen for TBM er en matematisk modell for TBM-prosjekter i harde bergarter. Modellen tar utgangspunkt i empirisk data fra tunnelprosjekter og beskriver hvordan forskjellige maskin- og geologi-parametere har en effekt på maskinens fremdrift og levetiden på kutterene. Den seneste iterasjonen av modellen ble utgitt i 2016 av Francisco Javier Macias i sin doktoravhandling "Hard Rock Tunnel Boring", nærmere bestemt vedlegget 'The NTNU prediction model for hard rock TBMs: Advance rate and cutter wear' [1]. Modellen presenterer sammenhenger mellom variabler med tabeller, funksjonsutrykk og grafer.

For å ta hensyn til hvordan geologien endrer seg gjennom fjellet gir NTNU-modellen formler for utregning per geologisk sone, der fjellet kan deles opp i flere etapper som har forskjellige verdier for de geologiske variablene. Det er også formler for å finne totalverdien for disse sonene, altså et samlet resultat for tunnelen. Prosjektrapporten "HARD ROCK TUNNEL BORING Advance Rate and Cutter Wear"av Amund Bruland gir flere formler for å beregne totalresultat, som gjennomsnittlig kutterlevetid, i sine vedlegg [5].

En sone kan i modellen inneholde én til tre sprekkesett. Et sprekkesett beskriver en til flere individuelle sprekkinger som er av lignende sprekkeklasse og har lignende vinkel i forhold til tunnelaksen. Sprekkeklasse er et mål på graden av oppsprekking i bergmassen [1, p. 364].

2.1.3 Modell-Parametere

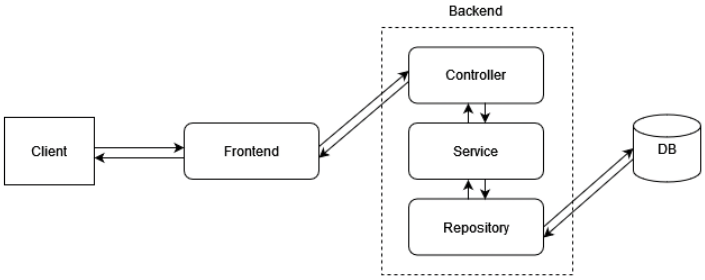
I en TBM refererer parametere til faktorer som påvirker ytelsen til TBMen i tunneldrivning. Disse parameterne kan omfatte parametere for bergartsforholdene, diameter på borehodet, matekraft på borehodet, borehodets hastighet, diameter på kutterdiskene, og skyvelengden på borhodet. Disse parametrene må tas i betraktning for å velge og konfigurere en TBM som passer best for et bestemt tunnelprosjekt, og for å optimalisere ytelsen til TBMen i løpet av boreprosessen. Parametere kan også justeres under boreprosessen for å sikre optimal ytelse i forhold til skiftende forhold i bergartene som blir boret gjennom. Eksempler på viktige parametere i NTNU TBM modellen er Drilling Rate Index (DRI), Cutter Life Index (CLI) og Orientation Of Fracture Set (OOFS). DRI er et indirekte mål på hvor mye arbeid som er nødvendig for å knuse stein og er dermed et effektivt mål for steinknusingprosessen til en kutter. CLI er et mål på en kutters levetid i borretimer. Til sammen utgjør CLI og DRI bergborbarhet, bergets evne til å bli boret. OOFS er et resultat av strøk- og fall-vinkelen mellom tunnelaksen og sprekken. OOFS referer til retningen og orienteringen av sprekker eller ledd i fjellmassen som TBM'en møter under utgravningen. Deres orientering kan ha betydelig innvirkning på utgravingsprosessen og stabiliteten til tunnelen. [1, p. 353]



Figur 2: DRI for forskjellige bergarter [1, Figur 2]

2.2 Arkitekturmønster CSR

CSR står for Controller-Service-Repository og er et designmønster for å strukturere kode i webapplikasjoner. I CSR er hver del ansvarlig for en spesifikk oppgave i applikasjonen. Controlleren håndterer inngående forespørsler fra brukere og sender dem til riktig Service. Controlleren håndterer også tilbakekall fra Service og avgjør hvordan dataene skal presenteres til klienten. Service har ansvaret for å håndtere virksomhetslogikk og koordinere interaksjonen mellom kontrolleren og databasen. Service er ofte delt opp i mindre tjenster for å håndtere spesifikke oppgaver. Repository har ansvaret for å kommunisere med databasen og utføre CRUD-operasjoner (Create, Read, Update, Delete). Fordelene med CSR er bedre struktur og organisering av kode, enklere vedlikehold og endringer i applikasjonen, og lettere å skille mellom ansvarområdene til hver del av koden.



Figur 3: Systemarkitektur med CSR på backend

2.3 Livsløpsvurdering

En livsløpsvurdering, eller Life Cycle Assessment (LCA), er en miljøvurdering på et produkt eller en prosess gjennom hele livsløpet, ofte beskrevet som 'fra vugge til grav'. De to påkrevde stegene i en slik vurdering er klassifisering og kategorisering av utslipp. Klassifiseringssteget går ut på å beregne totale utslipp av forskjellige avfallsstoffer, og kategoriseringssteget grupperer disse utslippene i kategorier baser på hva avfallstoffene bidrar til [6, p. 298]. De to relevante kategoriene for bærekraftsvruderingen i kapittelet Samfunnspåvirkning er 'Bidrag til klimaforandringer' (Climate change), og 'Menneskelig toksisitet' (Human toxicity).

2.4 Tidligere Arbeid

Fra tidligere finnes det to programvarer for NTNU-Modellen for TBM.

2.4.1 Anleggsdata Fullprof

Anleggsdata Fullprof er et program utviklet på 90-tallet for å beregne tid og kostnad for fullprofil boring av tunneler i hardt fjell. Programvaren er utviklet og markedsført i samarbeid med Construction Data og Institutt for bygg- og miljøteknikk ved NTNU. Beregningene er basert på empiriske data samlet inn av Ibaté ved NTNU. Programmet kan beregne kapasiteter som netto fremdrift, ukentlig fremdrift og nødvendig boretid, samt dreiemomentkontroll, skjæringsforbruk og normal og detaljert kostnad. Programmet gir beregninger basert på empiriske data og noen grunnleggende parametere, og brukerne kan justere mer enn 350 fakturaer basert på egne erfaringer. Programmet lar også brukerne lage diagrammer som viser forholdet mellom variasjoner i parametere, overstyre det vilkårlige detaljnivået og overvåke de valgte parametrene [7].

2.4.2 NTNU TBM 2017

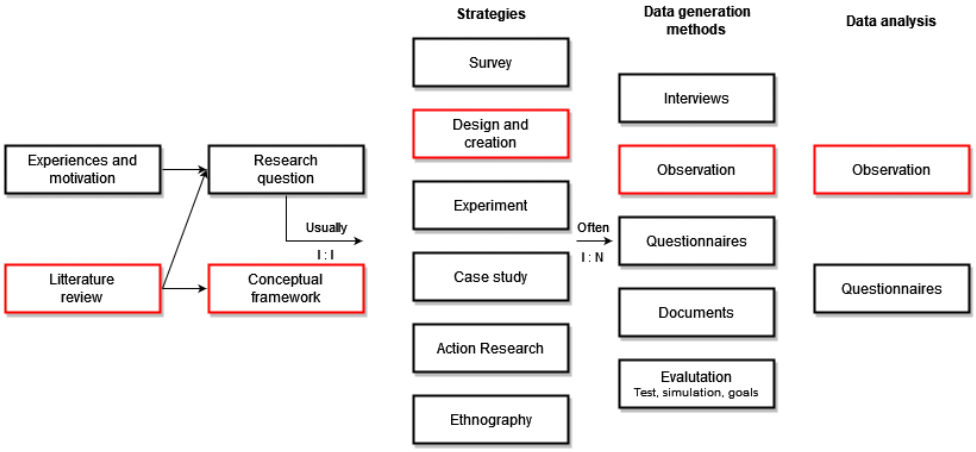
NTNU TBM er en videreutvikling av Fullprof utført i 2017 som en del av en bacheloroppgave ved NTNU. Oppgaven er basert på Amund Bruland og Javier Macias' doktorgradsavhandlinger (Hard Rock Tunnel Boring) samt programvaren Anleggsdata Fullprof. Sluttproduktet består av en webapplikasjon der brukeren kan kalkulere for eksempel: inndrift, kutter levetid, tid, samt flere utregninger som er relevant for best mulig planlegging ved tunnelboring. Teknologier brukt for å utvikle produktet var blant annet NodeJS, Express, Jquery og Pug [8].

3 Metode

Metodedelen beskriver forskning- og utviklingsmetode som ble brukt gjennom oppgaven. Forskningsmetode består av vitenskapelige metoder, og utviklingsmetode består av prosess- og teknologivalg for utviklingen.

3.1 Forskningmetode

Forskningsmetode er en systematisk tilnærming til å samle, analysere og tolke data for å besvare et spesifikt forskningsspørsmål eller å teste en hypotese. Forskningsmetode kan omfatte ulike tilnærminger, for eksempel kvantitative eller kvalitative. Kvantitative metoder innebærer vanligvis tallbaserte data og statistisk analyse [9], mens kvalitative metoder kan omfatte intervjuer, observasjon, og analyse av skriftlig materiale for å beskrive og forstå fenomener fra deltakernes perspektiv [10]. Forskningsmetode er viktig fordi den sikrer at forskningen er systematisk og velbegrunnet, og at resultatene kan anses som pålitelige og gyldige. Ved å følge en forskningsmetode, kan forskere unngå å trekke konklusjoner basert på antakelser eller personlige meninger, og i stedet stole på empiriske data og objektive analyser for å besvare forskningsspørsmålet. Dette bidrar til å opprettholde kvaliteten og integriteten til forskningen og kan ha betydelige konsekvenser for praksis, politikk og beslutningsprosesser.



Figur 4: Valgte metoder (basert på Fig.3.1 Model of the research process [11, p. 33])

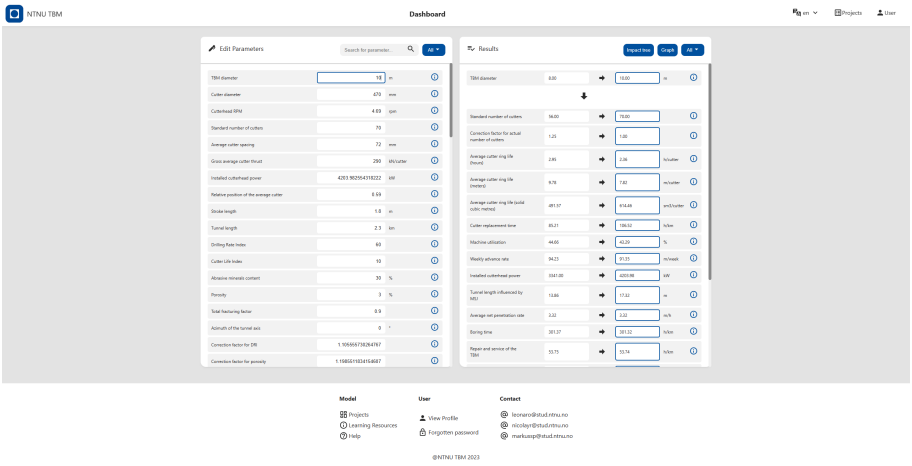
Siden problemstillingen er å digitalisere NTNU-modellen har det vært nødvendig å sette seg inn doktoravhandlingen som beskriver modellen i form av en litteraturgjennomgang. Med en grunnleggende forståelse for kildematerialet kunne gruppen skissere et konseptuellt rammeverk for hvordan den matematisk modellen burde oversettes til programvare. Strategien som ble utvalgt for å gjennomføre oppgaven var å jobbe iterativt for å kontinuerlig motta tilbakemeldinger fra oppgavestiller, slik at punktene 'Strategies', 'Data generation methods', og 'Data analysis' fra Figur 4 ble repetert flere ganger gjennom utviklingsperioden. For hver iterasjon ble en ny versjon av produktet utviklet og vist frem til oppgavestillerne. Gruppens metode for å generere og analysere data var gjennom møter med oppgavestillerne, hvor de fikk mulighet til å bruke programmet selv og gi tilbakemeldinger.

3.2 Utviklingsmetode

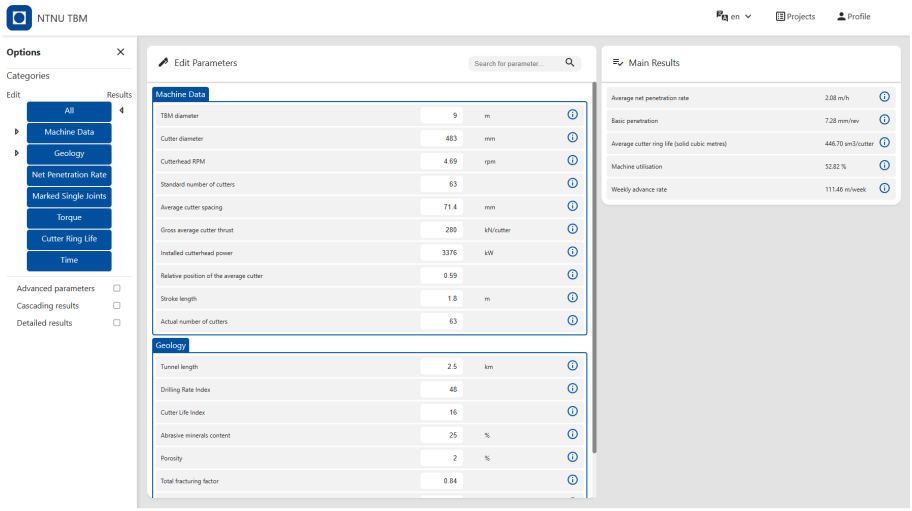
3.2.1 Prototyping

Gruppen har benyttet prototyping som en sentral utvikligsmetode. Formålet med prototypingen er å teste produktets funksjonalitet, design og brukervennlighet i en tidlig fase av utviklingsprosessen, slik at eventuelle feil og mangler kan oppdages og løses før den endelige versjonen av produktet utvikles. Prototyping har vært en verdifull metode for utvikling av løsningen, siden dette har gitt en mulighet til å teste ulike ideer og konsepter på en rask og enkel måte med produkteier underveis i prosjektet. Det har dermed vært enkelt å implementere brukerflyt tidlig i utviklingsfasen. En annen fordel ved prototyping er besparing av tid siden prototypene allerede er implementert i kode, i motsetning til wireframes som kun er skisseringer.

Videre har prototypene blitt dokumentert gjennom skjermdumper for hver sprint. Gruppen har da kunnet se utviklingen av komponentene gjennom utviklingsperioden for å få en oversikt over i hvilke retning utviklingen foregår, hva som har blitt endret og hvorfor og hvordan utviklingen kan fortsette videre for å oppnå best mulig resultat.



(a) Dashboard etter sprint 1



(b) Dashboard etter sprint 2

Figur 5: Utvikling av dashboardet ved hjelp av prototyping

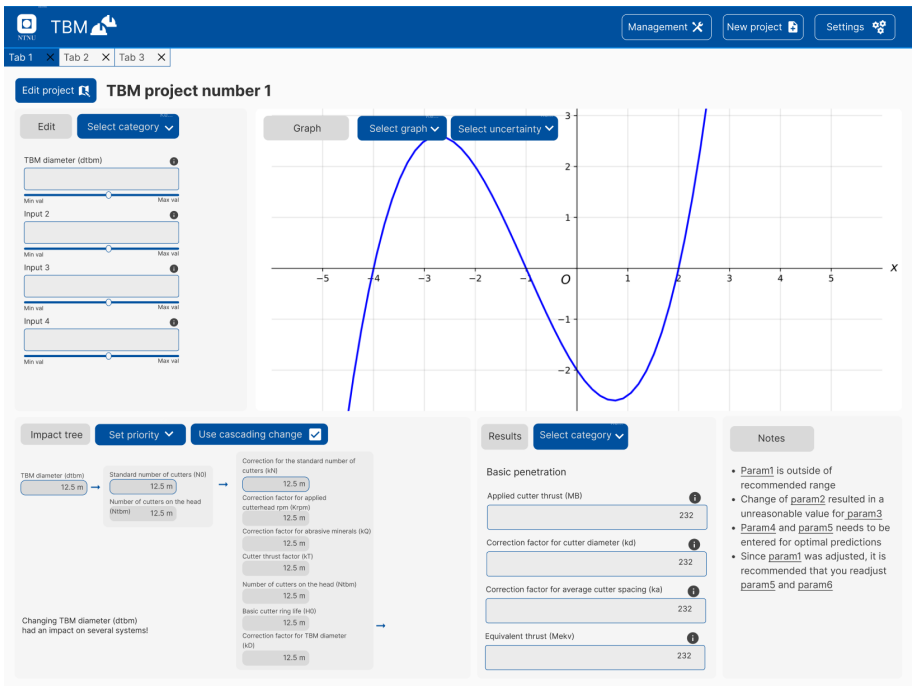
3.2.2 Smidig Utvikling Med Scrum

Prosjektarbeidet har foregått som en smidig utviklingsprosess inspirert av prosjektrammeverket Scrum. De to viktigste kriteriene for valg av prosess var at utviklingen skulle være iterativ og at den foregikk tett opp mot produkteier. En iterativ prosess sørger for at man til en hver tid har tilgang på et fungerende produkt, og gir mulighet for hyppig tilbakemeldinger. Dette er spesielt viktig i denne sammehengen, hvor en applikasjon skal utvikles fra bunnen av på begrenset tid. Utviklingsprosessen ble delt i flere sprinter på rundt to arbeidsuker. Hver sprint begynner med et planleggingsmøte hvor ønsket funksjonalitet prioriteres av produkteier. Disse punktene blir så delt i konkrete underpunkter som kan jobbes på av samtlige gruppemedlemmer. Ved slutten av sprinten gjennomgås resultatene sammen med produkteier, etterfulgt av et nytt planleggingsmøte og en ny sprint.

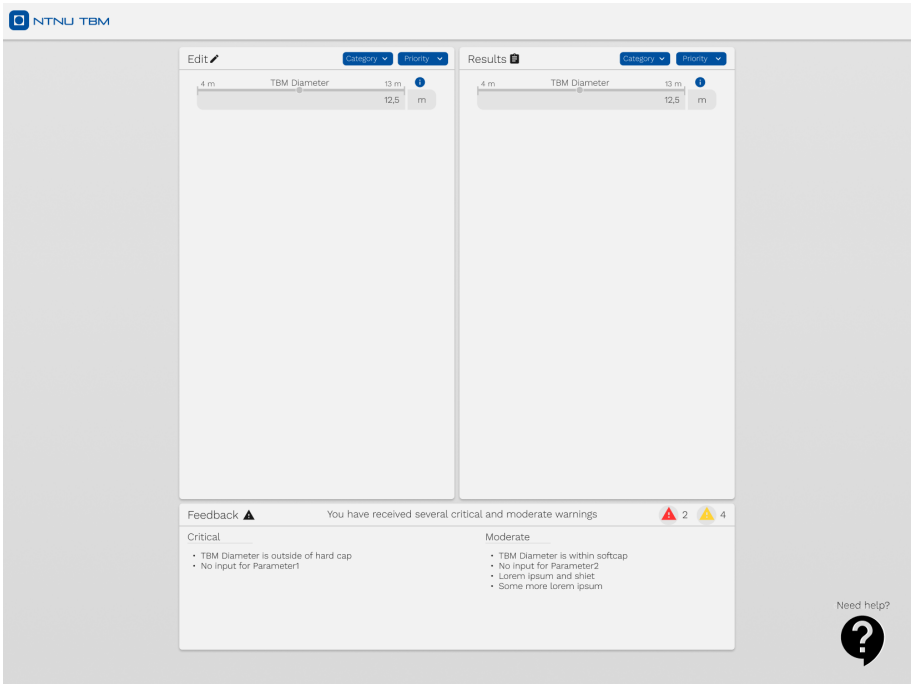
Gruppen har benyttet verktøy for revisjonshistorikk og issueboard for å fasilitere for enklere samarbeid på et delt prosjekt. Dette samler kildekoden på et felles sted, og fungerer som en svært viktig del av kvalitetsikringen på produktet. Funksjonalitet blir implementert på egne instanser av kildekoden, også kjent som feature branching, og må gjennomgås og godkjennes av minst én annen på gruppen før den sammenstilles med hovedkildekoden. I tillegg ligger oppgavene fra Sprint Backloggen i et online issueboard, der det kan visualiseres hvilket gruppemedlem som jobber på hvilken oppgave.

3.2.3 Wireframing

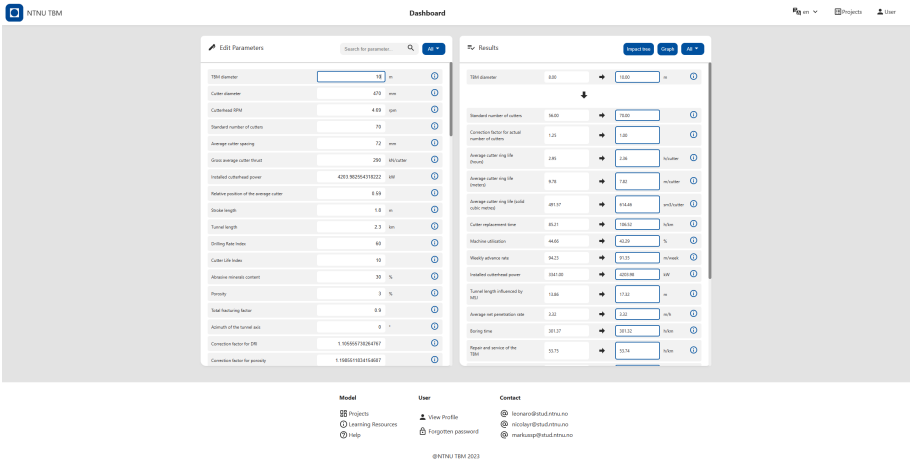
Wireframing er en teknikk som gruppen har brukt til å planlegge og visualisere hvordan nettsiden ville se ut og fungere før den ble utviklet, og gruppen har brukt wireframing i den initielle fasen av prosjektet. Gruppen har brukt wireframes til å planlegge layouten av nettsiden, inkludert hvor innhold og funksjoner skal plasseres på siden og hvordan de skal interagere med hverandre. Ved å lage flere wireframes, har gruppen sammenlignet og vurdert de ulike alternativene før valget om design som fungerer best for prosjektet har blitt gjort. Videre har wireframing vært til nytte for gruppen med å kommunisere designideene sine gruppemedlemmer imellom og med produkteier. Til slutt har wireframing også blitt brukt til å teste brukervennligheten til nettsiden i et tidlig stadium.



Figur 6: Dashboard wireframe versjon 1



(a) Dashboard wireframe versjon 2



(b) Dashboard etter sprint 1

Figur 7: Valgt design sammenlignet opp mot dashboardet etter sprint 1

3.2.4 Arbeids- Og Rollefordeling

Arbeids- og rollefordelingen kan kjennetegnes ved at den har vært fleksibel. Alle i gruppa har hatt en liknende motivasjon og kunnskapsnivå, som legger til rette for at et hvert medlem kan ta på seg de oppgavene som trengs, når det trengs. Dette resonerer med smidig utviklingsmetodikk: Dersom man blir ferdig med det man jobber med kan et hvilket som helst gruppemedlem ta på seg det neste punktet på listen. Dette sørger for å holde effektiviteten høy. Likevel tenderer forskjellige gruppermedlemmer til forskjellige arbeidsoppgaver, og ønsker/preferanser brukes også for å fordele oppgavene. Tanken er at hvis man jobber med noe man har lyst til er det morsommere og mer motiverende, som igjen kan føre til et bedre resultat.

Det hadde også vært mulig å ha mer definerte roller. Det er ofte hensiktsmessig å ha en leder som har ekstra ansvar for at gruppen følger prosessen, styrer møter, og å dra med seg hele gruppen. Gruppen valgte likevel ikke å ha én definert leder. Både størrelsen på gruppen, et svært tett samarbeid, og tidligere erfaringer med gruppearbeid tilsier at det ikke er behov i denne sammenhengen. Samtidig

finnes det situasjoner hvor det er behov for en leder, for eksempel under møter, og i slike situasjoner kan en møteleder utpekes ved møtestart.

3.3 Valg Av Teknologi Og Utviklingsmetode

Prosjektet er delt inn i en frontend nettside laget med Vue og Typescript, en backend server laget med Spring Boot i Java og en MySQL relasjonelldatabase.

3.3.1 Typescript

TypeScript og JavaScript er begge programmeringsspråk som brukes til å utvikle webapplikasjoner. JavaScript er et av de mest brukte programmeringsspråkene og brukes både på frontend og backend. Det er et dynamisk språk som kjøres direkte i nettleseren eller på en server ved hjelp av Node.js. JavaScript er enkelt å lære og kan brukes til å lage komplekse applikasjoner. TypeScript er et superset av JavaScript og legger til flere funksjoner og egenskaper. Det er et typet språk som betyr at variabler, funksjoner og objekter må deklareres med en bestemt datatype.

Dette prosjektet stiller krav til robusthet og høy kodekvalitet for visning av rett informasjon, gjenbrukbar kode og best mulig produktivitet ved utvikling. TypeScript gir muligheten til å definerer typer for variabler, funksjoner og objekter. Dette gjør det enklere å oppdage feil og øker kodekvaliteten. Videre økes produktiviteten ved utvikling fordi Typescript tilbyr IntelliSense i IDE (Integrated development environment). TypeScript øker også muligheten for gjenbruk av koden fordi typer kan defineres og brukes om igjen gjennom hele kodebasen.

3.3.2 Vue

Ved valg av frontend-rammeverk, er det flere alternativer å vurdere. De tre mest populære er Vue, React og Angular. Vue er et rammeverk som er enkelt å lære på grunn av en enkel syntaks, lett integrering med andre biblioteker, god dokumentasjon, høy popularitet og stort utviklersamfunn. React er et svært populært rammeverk som brukes av mange store selskaper blant annet Meta og Netflix. Det er enkelt å bruke og har et stort og aktivt samfunn som jobber med å utvikle nye tillegg og funksjonaliteter. React er også veldig fleksibelt og kan brukes til å bygge mange forskjellige typer applikasjoner. Angular er et mer omfattende og komplekst rammeverk som gir mange avanserte funksjoner og verktøy for å bygge store og komplekse applikasjoner. Det har en høy læringskurve, men gir også mye kontroll og fleksibilitet. Hver av disse rammeverkene har sine fordeler og ulemper, og valget har blitt gjort basert på kravene og behovene til prosjektet.

Behov og krav for dette prosjektet er ferdighetsnivå, ytelse, kompleksitet, og utviklingstid. En av hovedfordelene med Vue er at det blir sett på som enklere lære og komme i gang med sammenlignet med React og Angular. Vue er kjent for å ha en lettleselig syntax som er enkel å jobbe med og vedlikeholde. Vue har også god dokumentasjon og et stort og voksende utviklersamfunn, som gir mye støtte og ressurser. Vue har også bedre ytelse på DOM (Document Object Model) manipulering, oppstartstid og minnetildeling [12]. Videre har Vue oversiktelig filstruktur på grunn av at HTML, CSS og Typescript samles i én enkel vue-fil. React har seperate CSS og HTML/Javascript filer og Angular har seperate CSS, HTML og Javascript filer. Til større et prosjekt blir, til viktigere blir det å organisere koden. Ved å samle CSS, HTML og Typescript i samme fil løser vi dette problemet på en god måte.

Vue har blitt valgt til dette prosjekt til fordel for React og Angular fordi Vue har en enkel læringskurve, stort utviklersamfunn, god tilgang på oppdatert dokumentasjon, god ytelse og filer som samler all kode.

3.3.3 MySQL

Grunnen til at prosjektet tar i bruk relasjonsdatabasen MySQL fremfor andre løsninger som NoSQL-databaser eller andre relasjonsdatabaser som Oracle og Microsoft SQL Server er først og fremst fordi MySQL er gratis og er en åpen kildekode-database, som gjør den til et attraktivt alternativ for små og mellomstore bedrifter eller prosjekter med begrensede budsjetter. Videre er MySQL også en veldokumentert database med et stort og aktivt fellesskap av utviklere og brukere som tilbyr støtte og bidrar til utviklingen av databasen. Dette gjør det enkelt å finne løsninger på problemer og få hjelp når det trengs. MySQL er også en pålitelig og skalerbar database som kan håndtere store mengder data og høy trafikk. Den støtter også mange avanserte funksjoner og har et bredt utvalg av integrasjonsmuligheter med andre programvareverktøy og utviklingsrammeverk. Det er også verdt å nevne at MySQL er en av de mest populære relasjonsdatabasene i verden, og brukes av mange store selskaper og organisasjoner som Meta, Twitter og Google. Dette gjør det til en trygg og pålitelig database å velge for mange typer prosjekter [13].

Når det gjelder å velge en relasjonsdatabase fremfor en NoSQL-database, avhenger dette av prosjektets spesifikke krav og behov. Relasjonsdatabaser som MySQL er mer egnet for prosjekter med strukturerte data og komplekse spørringer, mens NoSQL-databaser er mer egnet for prosjekter med ustrukturerte data og behov for horisontal skalering. Siden dette er et prosjekt som krever tydelige relasjoner mellom entiteter egner relasjonsdatabasen MySQL seg godt.

3.3.4 Java Og Spring Boot

Java blir brukt som server-side programmeringsspråk i dette prosjektet. I et komplekst system med brukerer, prosjekter og en stor matematisk modell er man avhengig av en godt strukturert og pålitelig server. Siden frontend utvikles seperat eksponeres serveren gjennom et RESTful API. I tillegg må dataen kunne lagres til en database. Java dekker disse behovene svært godt.

For det første er et Java objektorientert språk. Dette passer spesielt godt for et stort system med strukturt data som skal bearbeides, lagres og oppdateres, og hentes. Med objektorientert kode man beskrive systemet med forståelig abstraksjoner slik som Prosjekt, Bruker, Parameter osv. Dette oversettes også veldig lett til datalagringen.

For det andre har Java et stort og aktivt miljø, og er mye brukt i industrien, som fører til gode tilgang på dokumentasjon og andre hjelperessurser. I tillegg er det utviklet flere nyttige rammeverk rettet mot spesifikke bruksområder. Et av disse er Spring Boot, som gjør det svært lett å sette opp REST APIer og å kommunisere med SQL-databaser gjennom Hibernate/JPA.

Siden serveren vil inneholde en del matematiske beregninger, må man også ta ytelse med i bilde. Programmeringsspråk som Rust og C++ er spesielt kjent for sin ytelse, men disse språkene brukes mere i systemprogramvare enn for REST-APIer og skriving til database, og har dermed ikke like god støtte for dette som Java hvor dette er en av de primære bruksområdene. Kravene til ytelse er ikke stor nok til at det veier opp for hvor tilrettelagt Java er for bruksområde.

Det finnes andre språk som er spesielt gode å lage REST APIer i slik som Python og JavaScript, med rammeverk som Django eller Flask for Python og Express for JavaScript. Til gjengjeld er disse språkene tolkede språk, som gjør ytelsen dårligere enn et kompilert språk som Java. Selv om språkene er enklere å skrive, spesielt Python som har svært leslig syntax, er Java med sin statisk typing å anse som mer passende et såpass komplekst system som dette.

Til slutt må Java og Spring Boot stilles opp mot C# og .NET, som er svært lignende og som dekker alle de samme behovene som Java. Den viktigste årsaken til å velge Java er plattformstøtte. .NET rammeverket er optimalisert for Windows, men applikasjonen skal publiseres til en lokal Linux-maskin. I tillegg har en av gruppemedlemmene Mac. Alt i alt er Java det mest passende valget.

3.3.5 Flyway

Ved en iterativ utviklingsmetodikk er det fordelaktig med et migreringsverktøy slik at databasen, både lokalt for utviklerne og i produksjonsmiljøet, ikke trenger å tilbakestilles for hver endring i strukturen. I tillegg er produktet avhengig av mye initiell data i form av standardverdier for parametere og tekstverdier for norsk og engelsk, noe som enkelt kan legges inn i form av et migreringsscript.

Flyway har blitt valgt som databasemigreringsverktøy. Hovedargumentet for Flyway er at migreringsscriptene er i form av SQL-filer som blir kalt i spesifisert rekkefølge, mens en løsning som Liquibase bruker XML-filer. Å kunne skrive databaseoppdateringer i SQL gjør teknologien intuitiv og man unngår opplæring i formatet Liquibase forventer.

3.3.6 GitLab

Ved versjonskontroll og samarbeid på kodeprosjekter er det flere mulig alternativer. GitHub er en av de mest populære plattformene for versjonskontroll og samarbeid på kodeprosjekter. GitHub tilbyr mange av de samme funksjonene som GitLab, inkludert kodegjennomgang, feilsporing og integrasjoner med andre verktøy. Andre platformer er Bitbucket som tilbyr funksjoner som kodegjennomgang, feilsporing og integrasjoner med andre verktøy, GitKraken som er en desktop-basert Git-klient som gir et visuelt grensesnitt for versjonskontroll og samarbeid på kodeprosjekter og SourceForge som er en av de eldste plattformene for open source-prosjekter. GitLab tilbyr også muligheten til å kjøre en instans av GitLab på en egen server, som kan gi ekstra kontroll og tilpasningsmuligheter for større team eller organisasjoner. Dette prosjektet bruker GitLab som versjonskontroll fordi det gjør det enklere å samarbeide internt både med veileder på NTNU og kontaktpersoner hos Institutt for bygg- og miljøteknikk.

3.3.7 Biblioteker

Axios Axios er et JavaScript-bibliotek som brukes til å gjøre HTTP-forespørsler fra weblesere eller Node.js-applikasjoner. Vi bruker Axios fordi Axios har et enkelt og intuitivt API som gjør det enkelt å sende HTTP-forespørsler og håndtere svar. Videre så støtter Axios avbryting av forespørsler eller svar før de håndteres, noe som kan være nyttig for å legge til egendefinerte headere eller autentiserings-token. Til slutt gir Axios robust feilhåndtering.

Material Design Icons Applikasjonen trenger ikoner for brukeropplevelsen og Material Design Icons har blitt brukt til dette. I Kombinasjon med Vue kan ikoner enkelt importeres og brukes med HTML-tag i koden.

JUnit og Mockito Junit er et rammeverk for enhetstesting av Java-applikasjoner som gir muligheten til å sette opp automatisk testing av klasser. I kombinasjon med Mockito kan andre klasser mockes for å isolere testingen til én enkelt klasse. Per Mai 2023 er disse de ledende rammeverkene for enhetstesting og mocking ifølge Maven [14].

Swagger/OpenAPI Swagger, eller OpenAPI den nyeste versjonen kalles, brukes til å sette opp dokumentasjon av REST-endepunktene i applikasjonen. Dette inneholder statuskoder, beskrivelser og forventede datatyper.

Java JWT Java JWT er et bibliotek som tilbyr JWT funksjonalitet for Java applikasjoner. Dette brukes både til signering og verifisering av tokens. Java JWT er på listen over anbefalte JWT-biblioteker fra auth0. [15]

3.3.8 Organisering Av Klasser

Det finnes flere måter å organisere klasser på blant annet ved bruk av MVC (Model-View-Controller), MVP (Model-View-Presenter), MVVM (Model-View-ViewModel) og VIPER (View-Interactor-Presenter-Entity-Router). Vi har valgt designmønsteret CSR (Controller-Service-Repository) på grunn av at CSR gir ønsket abstraksjon, spesielt opp mot Spring Boot hvor dette designmønsteret er det foretrukne valget. Mappestruktur for klassene kan finnes i Vedlegg E: Systemdokumentasjon.

4 Resultater

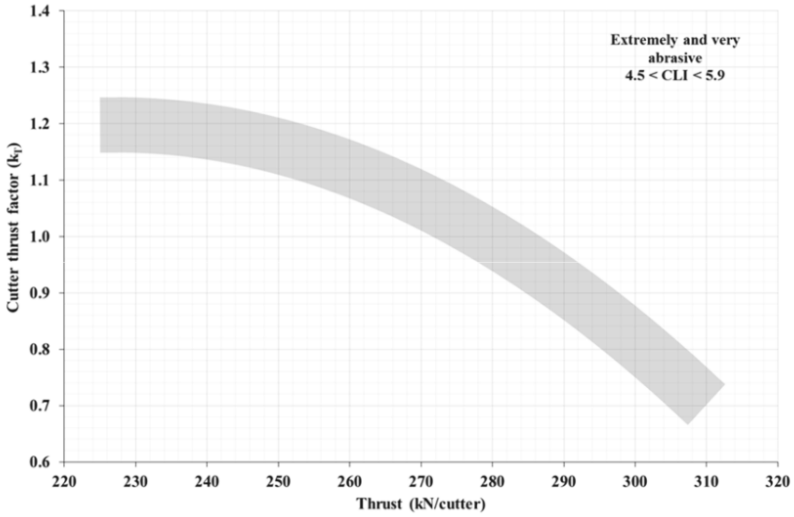
Resultatdelen beskriver de vitenskaplige og ingeniørfaglige resultatene som er oppnådd.

4.1 Vitenskaplige Resultater

Problemstillingen er i utgangspunktet ingeniørfaglig, og det ble ikke målsatt å komme frem til vitenskaplige resultater fra begynnelsen. Likevel har gruppen i sitt samarbeid med oppgavestillerne tatt i bruk videreutviklete formler fra en pågående doktoravhandling, samt introdusert egen videreutvikling underveis.

4.1.1 Digitalisering av videreutviklet modell

Frostad jobber med videreutvikling av NTNU-modellen i sin pågående doktoravhandling, og har i den sammenheng lagd et Excel-dokument for utregning [16]. Gruppen har fått tilgang til de nye formlene Frostad har utledet for å inkludere disse i produktet.

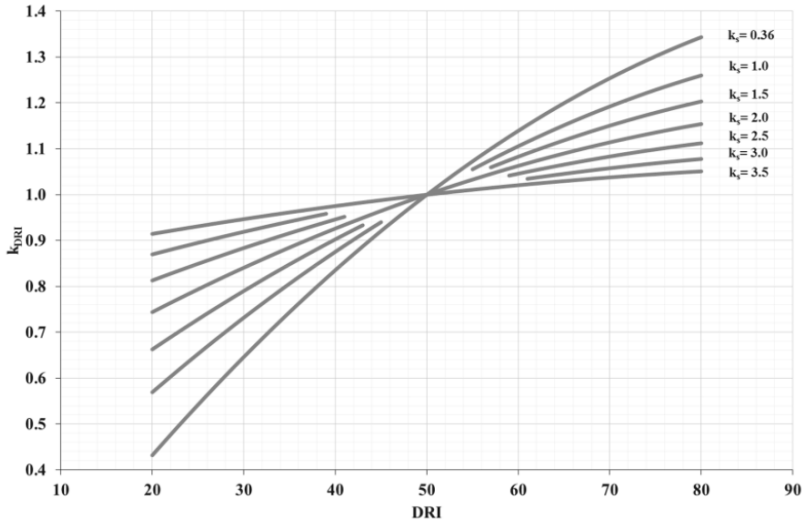


Figur 8: Korreksjonsfaktor for matekraft [1, Figur 27]

Sammenhengen mellom matekraft og dens tilhørende korreksjonsfaktor i den originale NTNU-modellen er definert gjennom grafen vist i Figur 8, der resultatet ikke har én nøyaktig verdi men et verdifelt. Frostad har kommet frem til spesifikke formler for denne sammenhengen, basert på CLI-verdien, og i tillegg utvidet definisjonsområdet til å inkludere CLI-verdier fra 4.5 til 12.0. Den nye metoden for å finne korreksjonsfaktor for matekraft er inkludert i produktet til gruppen. Det er i tillegg flere funksjoner hvor definisjonsområdet har blitt utvidet for forskjellige parametere.

4.1.2 Egen videreutvikling av modell

I samarbeid med Bruland og Frostad har gruppen også bidratt til videreutvikling av NTNU-modellen for TBM. I flere tilfeller er det oppgitt flere funksjoner for sammenhengen mellom to variabler, gitt en tredje variabel. Eksempelvis er funksjonen for korreksjonsfaktoren for DRI oppgitt for seks forskjellige sprekkefaktorer, vist i Figur 9. Sprekkefaktor er en kalkulert verdi som kan være kontinuerlig fra 0.36 til 3.5, og det er derfor interessant å finne en kontinuerlig funksjon for korreksjonsfaktoren. Dette er oppnådd ved å interpolere lineært mellom de oppgitte funksjonene fra den originale NTNU-modellen. Gruppen har introdusert lineær interpolering i alle tilfellene hvor et forhold er oppgitt i likt format som vist i eksempelet.



Figur 9: Korreksjonsfaktor for DRI [1, Figur 15]

Det er også introdusert til den digitaliserte modellen en kontinuerlig verdi for sprekkeklasse, som originalt er et heltall fra 0 til 7, som har gjort det mulig å interpolere flere funksjoner fra NTNU-modellen. I tillegg har flere formler blitt omgjort fra å bruke sprekkeklasse til å bruke sprekkefaktor, slik at den totale sprekkefaktoren for en sone kan brukes til resultatberegning istedenfor å beregne resultat for hvert enkelt sprekkesett.

4.2 Ingeniørfaglige Resultater

Dette underkapittelet beskriver resultatene vi har oppnådd opp mot funksjonelle og ikke-funksjonelle krav spesifisert i Vedlegg C: Visjonsdokument. Dette er henholdsvis brukersystem, prosjektsystem, NTNU-modellen for TBM, administratorpanel, Epost og språkstøtte og ikke-funksjonelle krav. Noen av målene har ikke blitt oppnådd, hvilket punkt dette gjelder blir informert om for hvert underkapittel.

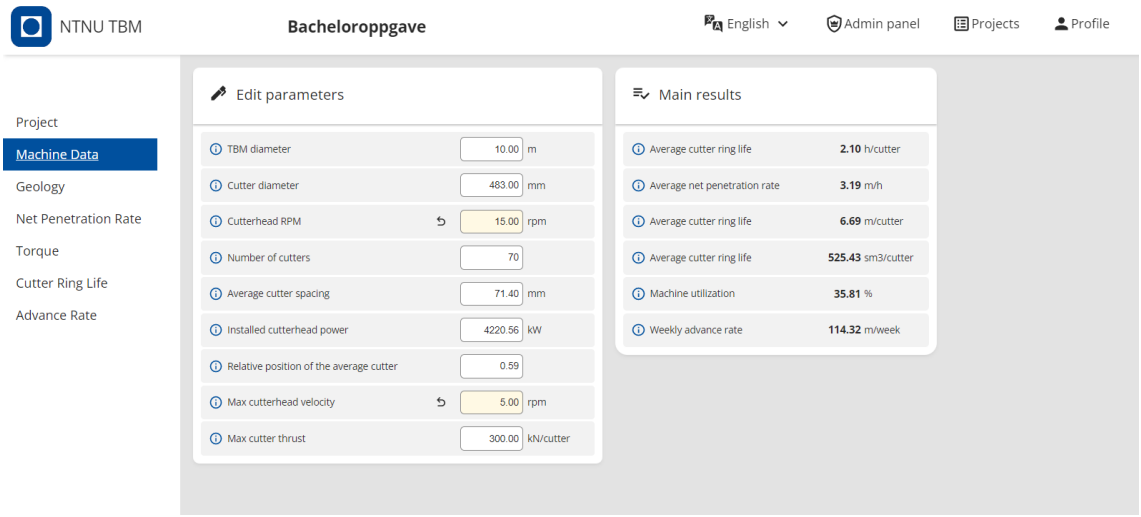
4.2.1 NTNU-Modellen for TBM

Alle matematiske uttrykk beskrevet i NTNU-modellen er lagt inn i systemet og validert med enhetstester. Dette inkluderer 81 parametere med 49 tilhørende matematiske funksjoner. Tjeneren har et system for endring av parameterverdier, der endringen forplanter seg i systemet slik det er beskrevet i NTNU-modellen. Hver parameter i systemet har felter for minimums- og maksimumsgrense som beskytter brukeren fra å skrive inn verdier som ikke er gyldig.

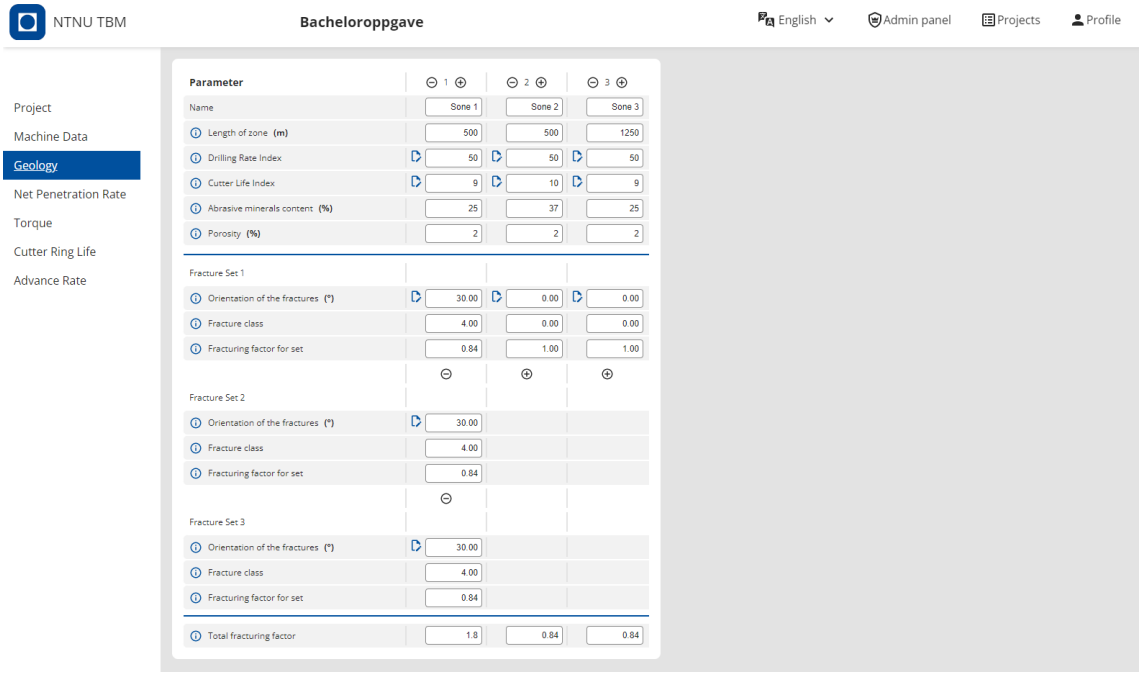
Det er mulig å dele tunnelen i flere soner, der geologiske verdier kan være forskjellige avhengig av hvor i tunnelen maskinen befinner seg. Hver sone får sine egne resultater, som fremdrift og kutterlevetid, og prosjektets resultater blir utledet fra de totale soneresultatene. For hver sone kan det defineres opp til tre sprekkesett, som beskrevet i modellen [1, p. 365]. Disse settene produserer en total sprekkefaktor for sonen, som har en betydelig innvirkning på maskinens fremdrift.

Parametere er sortert etter kategorier og prioriteringsnivå, for å angi i hvilke vindu de skal vises. Modellen representeres på dashboardsiden som brukeren får tilgang til ved å trykke på et prosjekt. I navigasjonen til venstre vises alle kategorier, og hvilken kategori som er valgt. Parametere kan vises på to format. Det første formatet i Figur 10 viser prosjektparametere for denne kategorien ved siden av 'Important' kategorien som vises på høyre siden. Det andre formatet er et soneformat vist i Figur 11. I dette formatet vises soneparameter og et utvalg av spesifiserte prosjekt-parametere. Videre har brukeren muligheten til å legge til et ubegrenset antall soner og maksimalt tre sprekkesett for hver sone. Øverst på siden står navnet til prosjektet.

Usikkerhet i modellen med Monte Carlo simulering har ikke blitt implementert.



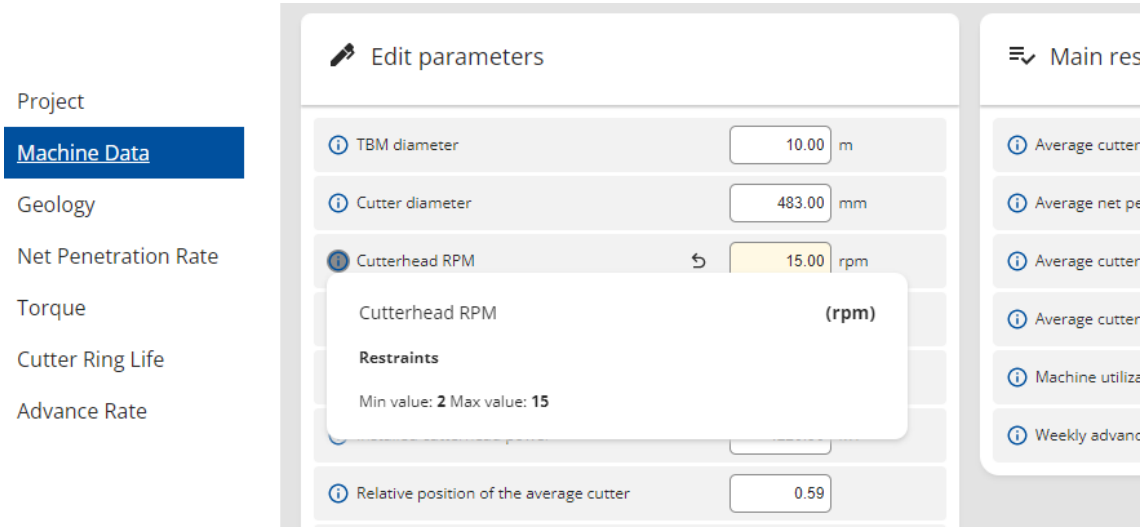
Figur 10: Brukergrensesnitt dashboard som viser kategorien 'Machine Data'



Figur 11: Brukergrensesnitt dashboard som viser kategorien 'Geology'

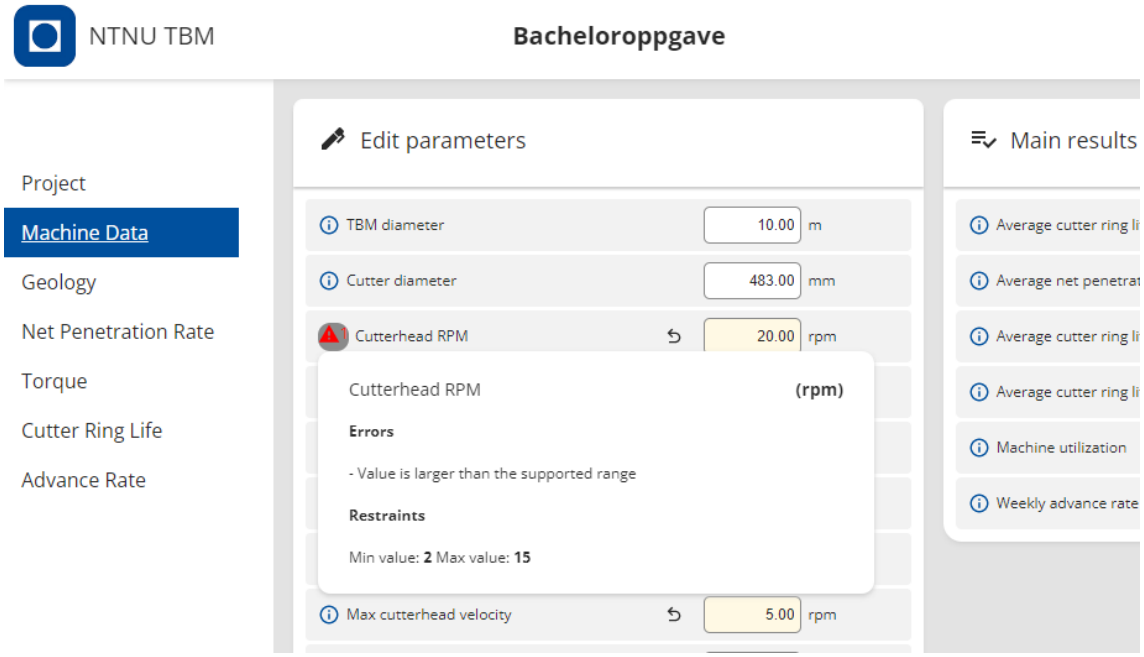
Hvert parameter har et informasjonsikon. Om brukeren trykker på ikonet vises en popup med informasjon om parameteret. Her vises navn, enhet, max og min verdi. Om parameter inneholder en beskrivelse, vises den her.

NTNU-modellen for TBM regner ut verdier for relevante parametere basert på hva brukeren legger inn av maskin- og geologi-parametere. I flere tilfeller vil brukeren gå over disse utregnede verdiene og overstyre disse. Når en verdi er overstyrt blir det visualisert i form av et tilbakestillings-ikon ved siden av parameterverdien. I Figur 12 har brukeren overstyrt 'Cutterhead RPM'. Ved å klikke på tilbakestillingsikonet vil verdien gå tilbake til modellens foreslåtte verdi.



Figur 12: Brukergrensesnitt parameter informasjon

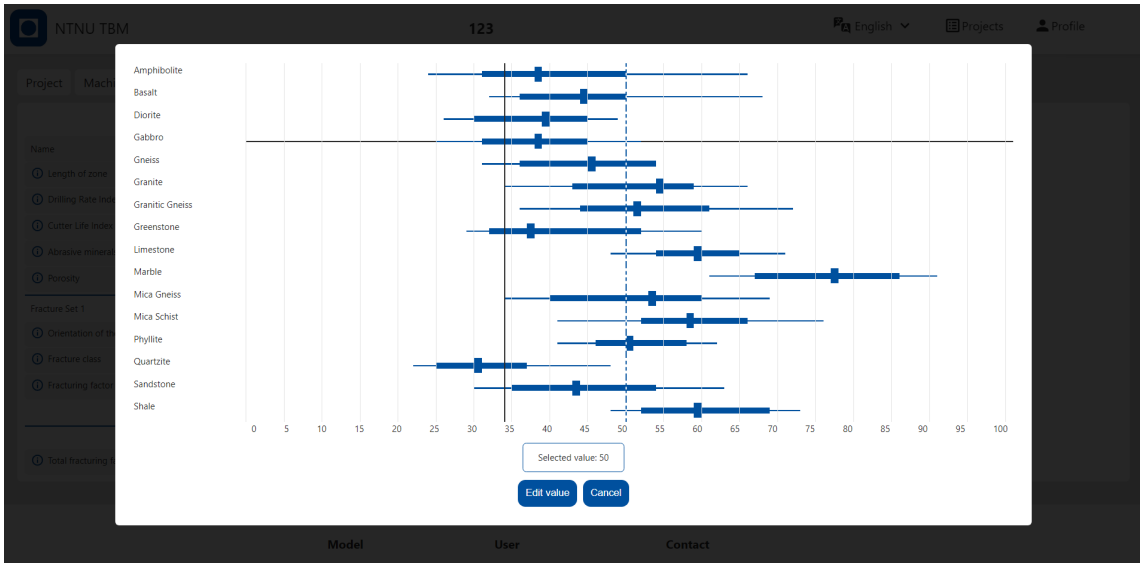
Om backend oppdager at inntastet verdi for parameter er utenfor gyldighetsområde sendes feilkoden 400: 'Bad request' til frontend. Frontend catcher denne feilmeldingen og kan generer en feilmelding basert på max- og minverdi for parameteret som vises til brukeren. Dette er med på å bevare integriteten til modellen siden dette motvirker at en ugyldig verdi kan sende andre parametere utenfor sitt gyldighetsområdet. Figur 13 viser at når det er en feil på parameter 'Cutterhead RPM' med en verdi på 20 blir informasjonsikonet endret til en varseltrekant som viser antall feilmeldinger. Minimal og maksimal verdi på dette parameter er angitt til å være 2 og 15, definert fra modellen [1, Figur 9, p. 361]. Verdien er dermed utenfor det tillate verdiområdet.



Figur 13: Brukergrensesnitt feil på Cutterhead RPM

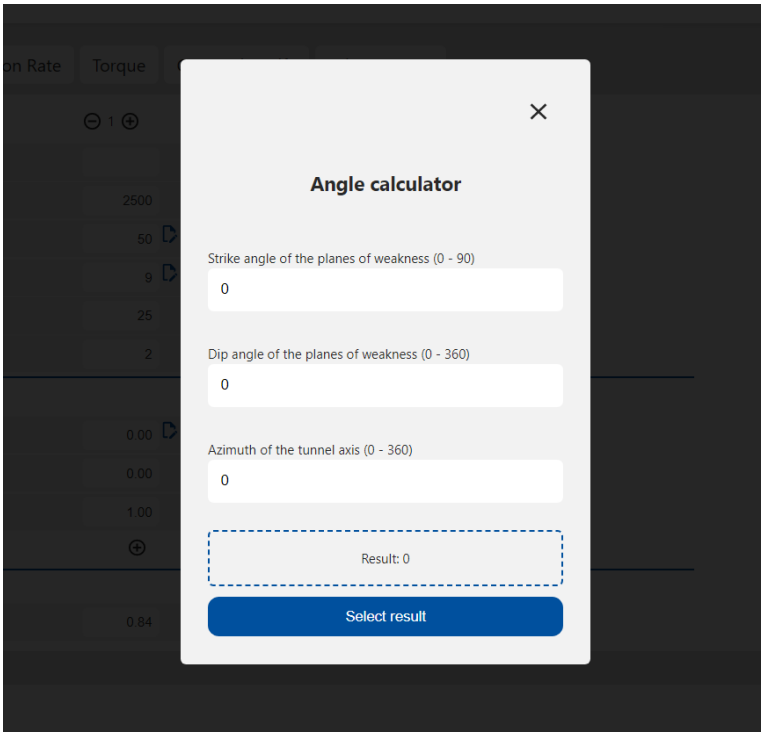
Input til parametere er tallverdier, hvor OOFS alternativt kan få tilegnet verdi gjennom vinkelkalkulator og verdi for DRI og CLI kan alternativt hentes fra horisontal boksplotgraf (Horizontal Boxplot chart). Boksplottet baserer seg på Figur 2 og 3 i Hard Rock Tunnel Boring [1, p. 354]. Boksplottet har blitt implementert gjennom bruk av Grid. Hver bergart har verdier for median, første kvartil, minimum- og maksimum-verdi. Tynne streker er området innenfor minimum og maksimum,

tykke streker er området innenfor første kvartil og den tykkest streken er medianverdien. Brukeren kan bruker muspeker for å velge verdi. Dette vises som et svart kryss. I Figur 14 kan krysset ses for bergarten Gabbro. Valgt verdi vises på bunnen av siden. Når knapp 'Edit value' trykkes på, sendes verdien tilbake til dashbordet for å bli brukt i utregning av den oppdaterte modellen. Systemer for steintyper er dynamisk. Det vil si at om et parameter har tilegnet steintyper, vil symbolet for å hente verdi fra bokspott automatisk dukke opp. Input fra dropdown er ikke implementert.



Figur 14: Brukergrensesnitt boksplott ved valg av DRI og CLI

Inputfelter til vinkelkalkulator for OOFS har begrensninger som er angitt ved siden av navnet på inputfeltet. Ved gyldige verdier for alle input-felt brukes formelen for OOFS [1, p. 365] til å regne ut resultat som kan sendes tilbake til dashbordet.

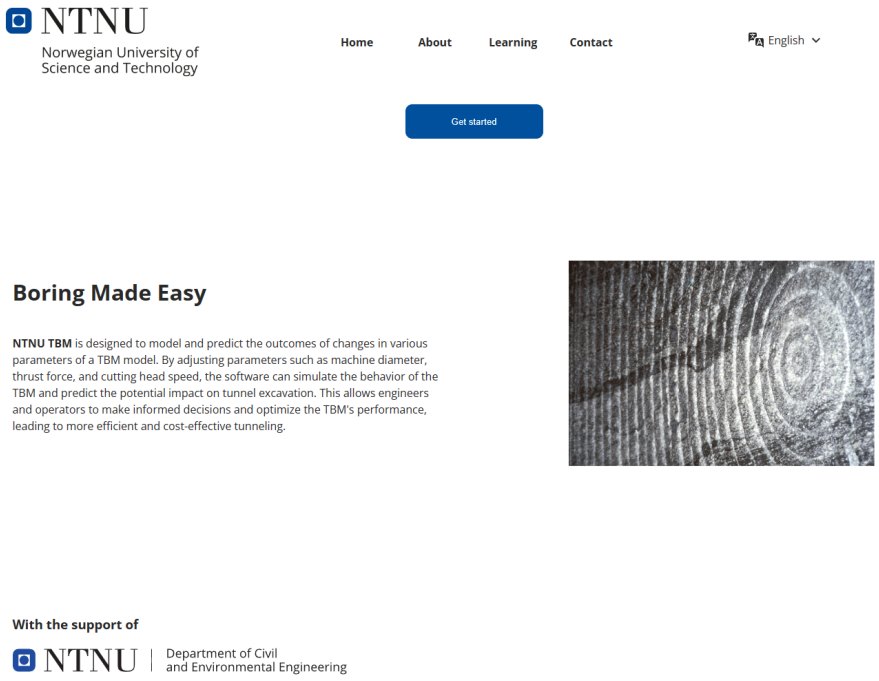


Figur 15: Brukergrensesnitt vinkelkalulator ved valg av OOFS

4.2.2 Brukersystem

Programmet har et brukersystem med landingside, opprettelse, endring og sletting av bruker, logge inn og ut, og navigasjon. I kan brukeren tilbakestille passordet sitt, og justere innstillinger som språk og fargemodus.

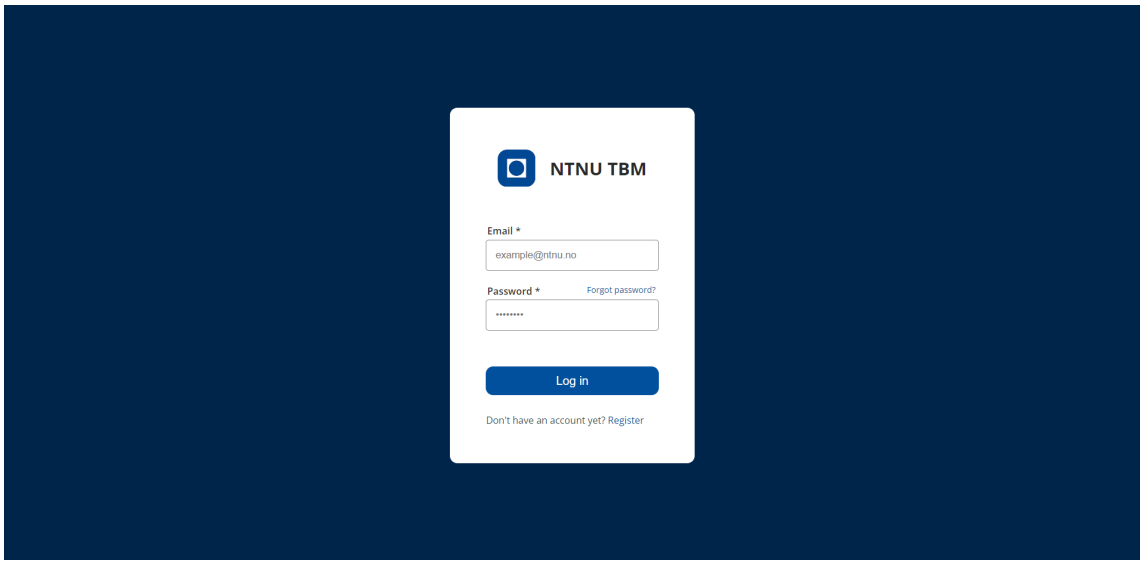
Landingside
Programmet har en landingside ment for personer som ikke har opprettet bruker. Siden består av en kort introduksjon til NTNU TBM, Hva NTNU TBM er, litt om TBM og en kontakt del. Fra landing siden kan brukeren bli omdirigert til logg inn/registrer siden. Om brukeren prøver å få tilgang til sider som er begrenset til innloggede brukere, blir brukeren omdirigert til landing siden. Det omvendte gjelder også. Om brukeren prøver å gå til landingsside som innlogget bruker, blir bruker omdirigert til prosjekt-siden.



Figur 16: Brukergrensesnitt for landing side

Logg inn
Logg inn siden gjør det mulig for en bruker å logge seg inn ved hjelp av epost og passord. Om felt står tomme bes brukeren om å fylle inn manglende felt. Passord-feltet vises som maskert istedet for klartekst. I tillegg vises en glemt passord knapp ved siden av passordet-feltet. Denne knappen omdirigerer brukeren til en glemt passord side. Her kan brukeren oppgi epost-adressen sin og motta en lenke hvor de kan angi et nytt passord. Fra logg inn siden kan man også komme til registreringsiden.

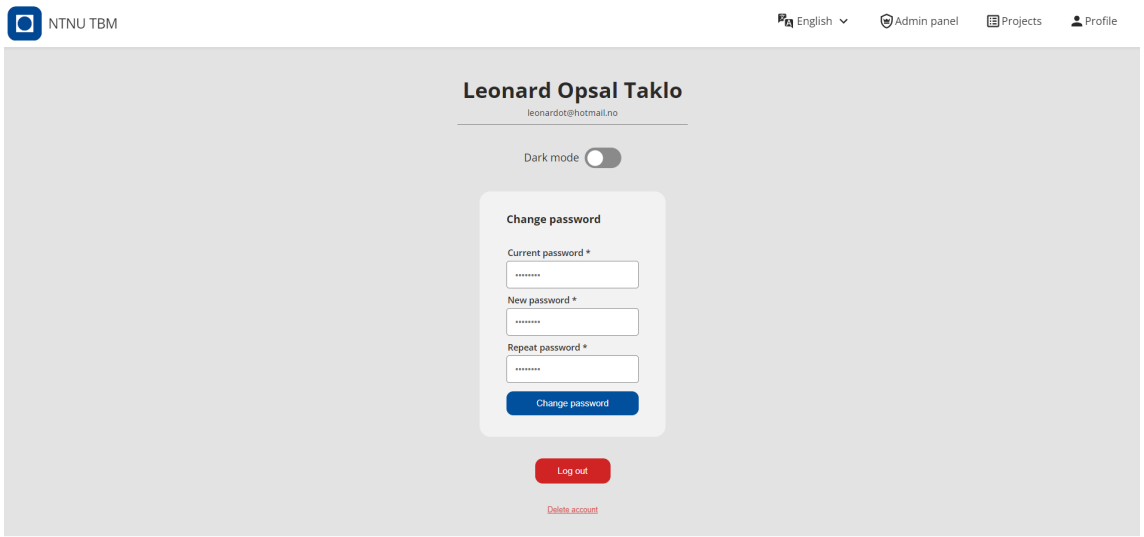
Når en logger inn omdirigeres man til prosjektsiden. Bak kulissene forblir brukeren pålogget ved hjelp av et omfattende token-basert system. Mer detaljer om dette finnes i Kapittel 4.2.7 Sikkerhet.



Figur 17: Brukergrensesnitt logg inn side

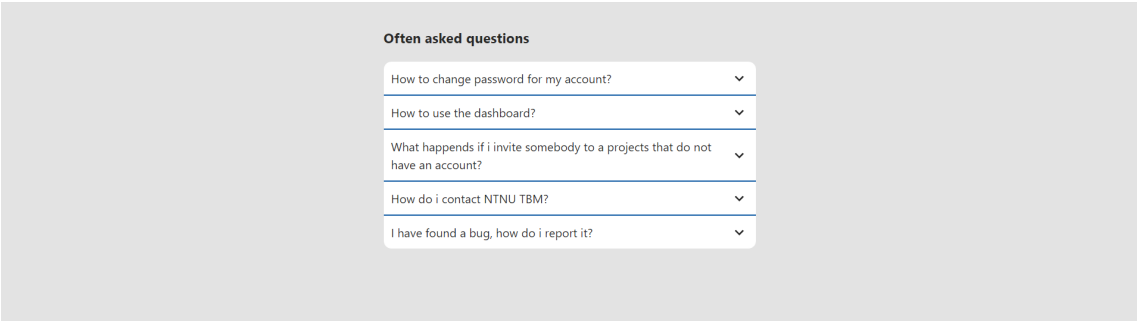
Registreringsside
Registreringssiden gjør det mulig for personer å opprette bruker i systemet. Informasjon som bes fra bruker er navn, epost, passord og passordet repetert. Passord og repetert passord er maskert. På denne siden er det også en link til side for brukervilkår samt omdirigering tilbake til logg inn siden. Etter brukeren har blitt registrert blir man omdirigert til verifiseringssiden hvor man blir bedt om å verifisere epost knyttet til brukeren.

Brukerside
Brukersiden viser informasjon om innlogget bruker som epost, innstillinger, mulighet til å endre passord, logg ut funksjonalitet og sletting av bruker. Bytting av passord vises som en egen boks under innstillinger. Under bytting av passord knapp er en logg ut knapp, som logger brukeren av systemet. Deretter omdirigeres brukeren til landingsiden. På bunnen av siden er det en knapp for å slette brukeren fra systemet.



Figur 18: Brukergrensesnitt brukerside

Hjelpeside
Nettsiden har en hjelpeside hvor brukeren kan få svar på det mest generelle spørsmålene. Det er fem spørsmål totalt. Når brukeren trykker på et spørsmål ekspanderes den for å vise svaret.



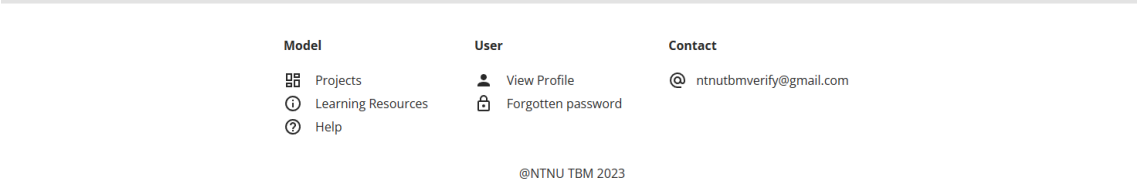
Figur 19: Brukergrensesnitt hjelpeside

Navigering mellom sider
Generell navigering blir gjort gjennom navigasjonen på toppen av siden og footeren på bunnen av siden. Navigasjon på toppen av siden gjør det mulig å bytte språk og navigering til prosjekt og brukersiden. Om brukeren er administrator vil også en admin panel knapp vises. Selv om brukeren scroller, vil navigasjons alltid vises på toppen av siden.

Figur 20: Brukergrensesnitt navigasjon som ordinær bruker

Figur 21: Brukergrensesnitt navigasjon som administrator bruker

I tillegg til navigasjon på toppen av siden, har siden en footer som viser lenker til prosjekt, læringsressurser og hjelpeside under modellkategori. Profilside og bytting av passord vises under brukerkategori og kontaktinformasjon under kontaktkategori. Footer er plassert helt nederst på alle sider utenom landing siden, for enklest mulig tilgang til navigering til applikasjonens tilgjengelige sider. Dette er også kjent som et sidekart.

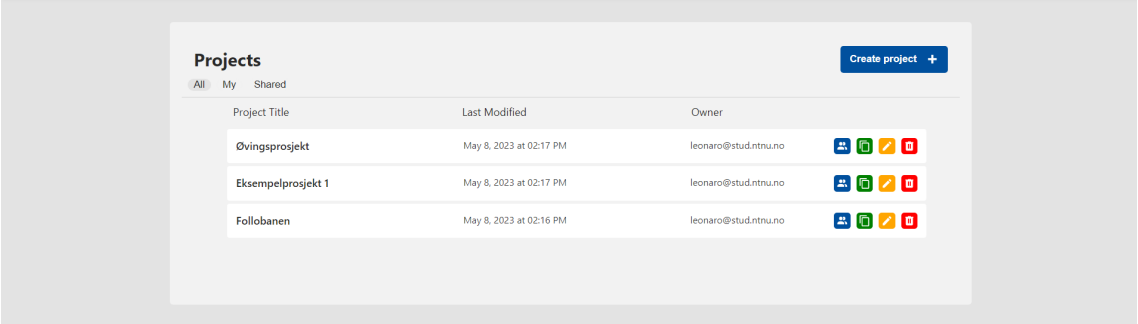


Figur 22: Brukergrensesnitt footer

4.2.3 Prosjektsystem

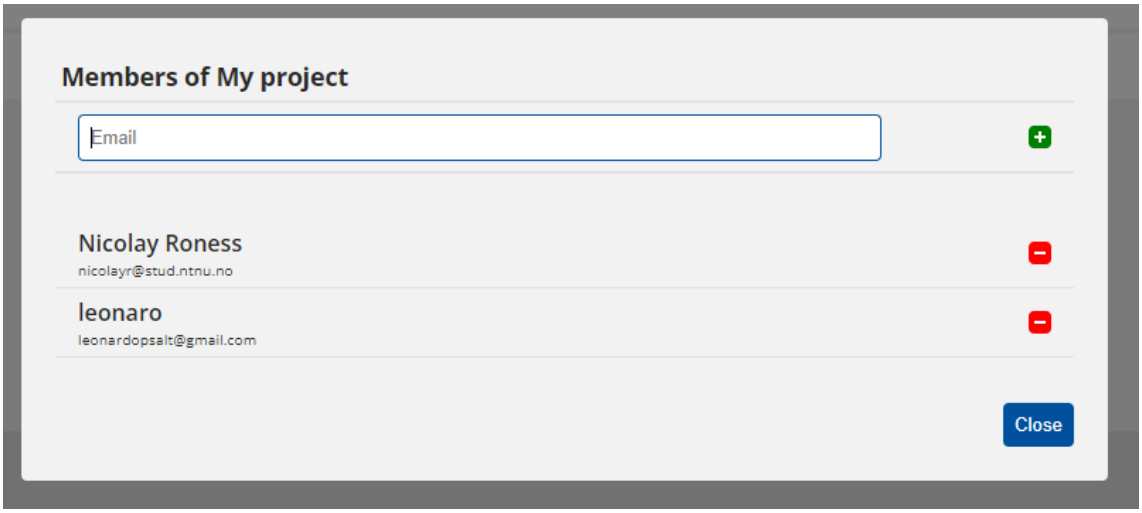
Et prosjekt er en instans av NTNU TBM-modellen. Innloggede brukere kan opprette nye prosjekter, som består av standardverdier ved opprettelse. Prosjektene organiser i lister hvor alle prosjekter vises. Dette inkluderer prosjekter brukeren selv har opprettet, og andre prosjekter brukeren er med i. Videre vises kolloner for navnet på prosjektet, tidspunkt for forrige endring i prosjektet, og

eposten til eieren av prosjektet. Utskrift av et prosjekt i PDF-format og endring av verdier i sanntid (WebSocket) har ikke blitt implementert. Eieren av prosjektet kan endre navnet på prosjektet, invitere nye medlemmer via epost eller fjerne medlemmer. Prosjektlisten er sortert basert slik at det prosjektet som sist ble endret vises øverst. Man kan også velge å sortere på prosjektnavn og eier. Videre kan man velge å se alle prosjekter, kun se egne prosjekter eller kun se prosjekter man er invitert til.



Figur 23: Brukergrensesnitt prosjektsiden

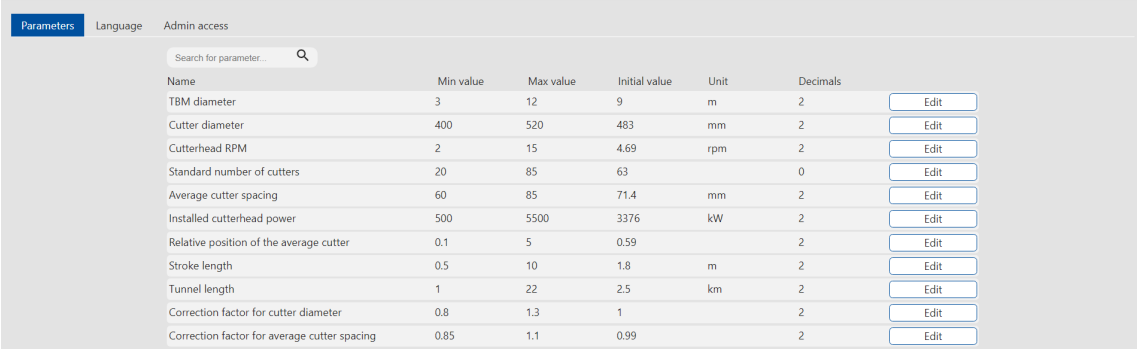
På et eksisterende prosjekt kan bruker invitere medlemmer, duplisere prosjektet, redigere navn og slette prosjektet. Inputfelt for å legge til medlemmer er øverst og liste over medlemmer i prosjektet nederst. Her har brukeren også muligheten til å fjerne medlemmer.



Figur 24: Brukergrensesnitt oversikt og invitasjon av medlemmer

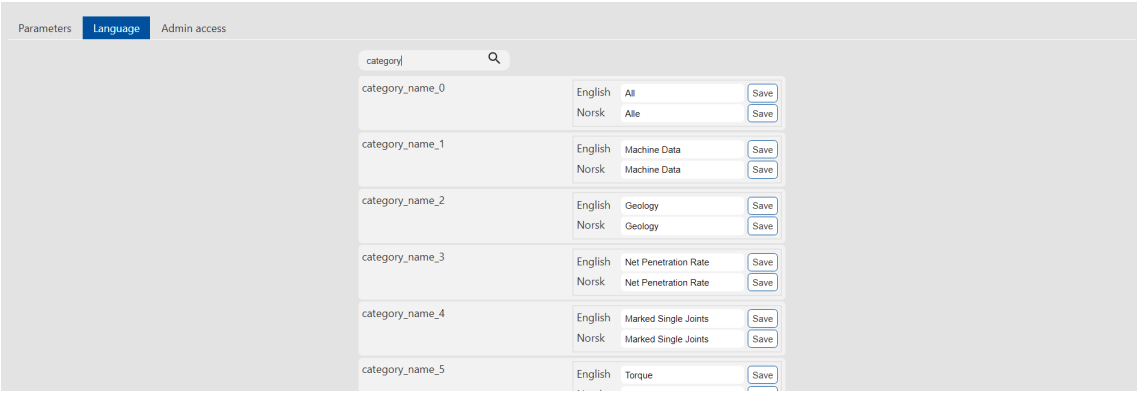
4.2.4 Administratorpanel

Som administrator har man adgang til administratorpanelet. Administrator-bruker kan navigere til panelet via administratorknapp i navgiasjonsmenyen på toppen av siden. Administratorpanel er kategorisert i parameter, språk og administrator-tilgang. I parameter kan bruker endre tallverdier knyttet til prosjektparameter, sone parameter og fracture set parameter. Verdiene som kan endres er minimuverdi, maksimumverdi, initiell verdi (standardverdi), enhet og antall desimaler som vises. Øverst for hver seksjon på parametersiden finnes det er søkefelt for å søke på navnet til parameteret. Sletting av bruker som administrator har ikke blitt implementert.



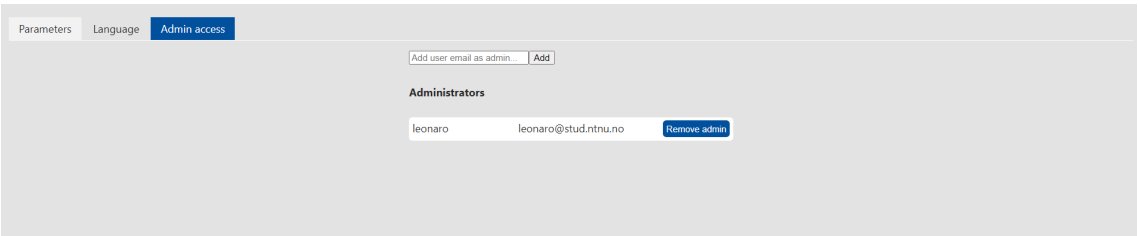
Figur 25: Brukergrensesnitt administrator parameter tab

I språk kan bruker endre tekst på språkene som systemet støtter. Her får man oppgitt alle nøklene som kan søkes på via søkefeltet øverst på siden. Søkefeltet kan søke både på nøkler og språkverdier for alle språk. Hver nøkkel har en tekst knyttet til seg for hvert språk.



Figur 26: Brukergrensesnitt språk tab

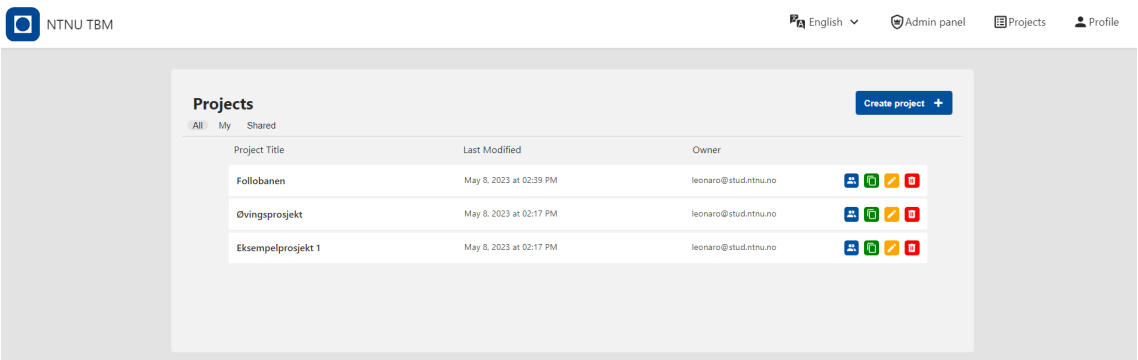
I administrator-tilgang får bruker en oversikt over alle administratorbrukere ved mulighet til å fjerne en bruker som administrator. Øverst kan bruker legge til andre administratorbruker ved utfylling av epost. Om epost ikke finnes gis det feilmelding.



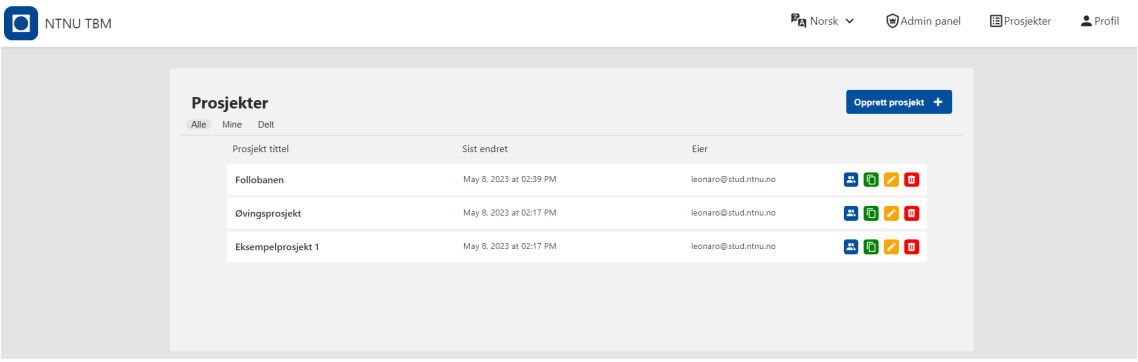
Figur 27: Brukergrensesnitt administratortilgang tab

4.2.5 Språkstøtte

Programmet støtter språkene engelsk og norsk. Valg av språk blir gjort i navigasjonsmenyen på toppen av siden og om person er logget inn blir språk lagret som innstilling for bruker. Administratorbruker kan endre språkverdier i administratorpanelet.



Figur 28: Brukergrensesnitt prosjektside med engelsk oversettelse



Figur 29: Brukergrensesnitt prosjektside med norsk oversettelse

4.2.6 Epost

Applikasjonen sender ut epost i tre tilfeller. Det første er ved registrering av en ny bruker. Da sendes en epost som inneholder en lenke som aktiverer brukeren. Man kan altså ikke logge seg inn på nettsiden før epostadressen er blitt godkjent på denne måten. Det andre tilfellet når en epost sendes, er ved tilbakestilling av passord. Dersom brukeren har glemt passordet sitt kan de be om få sette et nytt ved å skrive inn epostadressen sin. Her sendes en lenke hvor et nytt passord kan settes. Det siste tilfellet er når en bruker ønsker å invitere andre brukere til prosjektet sitt. Invitasjonseposten som sendes inneholder informasjon om prosjektet og en lenke hvor invitasjonen kan aksepteres.

4.2.7 Ikke-Funksjonelle Egenskaper

WCAG
Prosjektet skulle følge WCAG 2.1 som er en serie med web-tilgjengelighets rettningslinjer for uvikling av webapplikasjoner. Kontrastsjekker som har blitt brukt er fra WebAim [17]. 47 punkter fra WCAG 2.1 sjekklisten har blitt sjekket og resultatet er vedlagt i Vedlegg F: WCAG-sjekkliste. Videre er en oppsummering av resultatet:

Tabell 1: WCAG 2.1 punkter

Punkter	Støtte	Delvis støtte	Ikke relevant
47	37	1	9

Det er totalt 47 punkter i WCAG 2.1, og applikasjonen oppfyller 37 av kravene. Det er delvis støtte for tekststørrelse-justering i nettleser. Videre er det 9 punkter som ikke er relevant som hovedsakelig er bestående av video, lyd og bevegelsesaktivering.

Sikkerhet
Det finnes funksjonalitet i applikasjonen som har et direkte formål om å øke sikkerheten ved bruk av nettsiden, spesielt knyttet til brukersystemet. Det to viktigste tiltakene er knyttet til lagring av passord og å huske den påloggede brukeren. Passord blir både hashet og saltet før de lagres i databasen. For hashing benyttes *bcrypt*-algoritmen med en styrke som gjør at det tar omtrent 250-1000ms å hashe et passord i produksjonsmiljø. Hele prosessen drar nytte av API-et til Spring Security biblioteket. For å holde brukeren pålogget benyttes et token-basert system med en access token som lagres i minnet og en refresh token som lagres i en cookie. Teknisk spesifikasjon for dette blir beskrevet i Vedlegg E: Systemdokumentasjon.

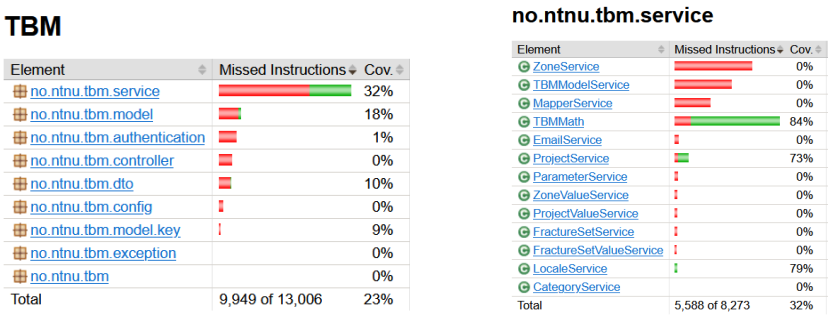
Vi tar utgangspunkt i OWASP TOP 10 listen fra 2021 over sikkerhetstrusler på nett [18], for å evaluere sikkerheten til nettsiden, og kartlegge hvilke sikkerhetstiltak som er nødvendig. Her følger en gjennomgang av listen:

- **A01-Broken Access Control:** Den mest kritiske og vanligste feilen i webapplikasjoner er ødelagt tilgangskontroll. Kort oppsummert går dette ut på at en bruker får tilgang til informasjon eller funksjonalitet den egentlig ikke skal. Applikasjonen vår inneholder flere tiltakt for å beskytte mot dette, hvorav det overnevnte tokensystemet er en viktig bidragsyter. Klienten sender automatisk vekk brukere fra sider de ikke skal ha tilgang til. Skulle man klare å forbipassere dette blir man stoppet på serversiden. Et eksempel på dette er at dersom man forsøker å endre en verdi i modellen sjekkes det at brukeren knyttet til tokenen faktisk er medlem i det prosjektet. Et annet eksempel er administratorsiden. Forsøker man å besøke den uten tilgang blir man sendt vekk. Forsøker man å gjøre API-kall som krever administratortilgang som vanlig bruker blir man også stoppet.
- **A02-Cryptographic Failures:** Kryptografiske feil kan forekomme dersom data som burde være kryptert ikke er kryptert, eller er kryptert dårlig. Det viktigste tiltakt i konteksten av dette produktet er å bruke HTTPS, slik at datatrafikk fra nettsiden er kryptert. Oppsett av HTTPS blir i dette tilfellet administrert av noen andre.
- **A03-Injection:** Dersom en applikasjon ikke er tilstrekkelig beskyttet mot injeksjonsangrep, kan en angriper kjøre vilkårlig, og potensielt ondsinnet, kildekode. Ved å benytte rammeverk som Spring Boot og Vue er produktet godt beskyttet mot injeksjon av SQL og Javascript (SQL-injection og XSS).
- **A04-Insecure Design:** En godt implementert funksjon, som baserer seg på et sikkerhetsmessig dårlig design. Det motvirkes i denne applikasjonen ved at implementasjoner basert på til anbefalinger respekterte aktører som Auth0 [19] og Spring [20].
- **A05-Security Configuration:** Feilaktige konfigurasjon kan føre til sikkerhetshull. Innstillinger i applikasjonen er satt opp i henhold til anbefalinger fra relevant dokumentasjon. Et eksempel på dette er at den hemmelige nøkkelen som brukes til signering av JWTs må være kompatibel med krypteringsalgoritmen som brukes for at den kan ansees som sikker.
- **A06-Vulnarable and Oudated Components:** Eldre eller mindre kjente kildekode, for eksempel fra biblioteker, har økt sannsynlighet for sikkerhetshull. Derfor er applikasjonen forsøkt implementert i henhold til moderne sikkerhetsanbefalinger. Eksterne biblioteker bruker nyeste versjoner, og eksempelvis biblioteket som brukes til å validere JWTs står på den offisielle listen over anbefalte biblioteker til dette.
- **A07-Identification and Authentication Failures:** Autentisering leder typisk til potensielle smutthull. Applikasjonen beskyttes mot gjennom omfattende tokensystem. Applikasjonen har ikke tofaktorautentisering og er dermed utsatt for 'brute force'-angrep av vanlig passord eller en stor mengde passordkombinasjoner. Et tiltak mot dette er at hashingalgoritmen er justert for å bruke omtrent 1 sekund på å sjekke et passord, som begrenser hastighet passord kan sjekkes under et eventuelt angrep.
- **A08-Software And Data Integrity Failures:** Denne feilen er typisk knyttet til ekstern eller utdatert kildekode. Applikasjonen har få eksterne avhengigheter. I tillegg kjøres produksjonskoden i et dedikert miljø. Evt. tredjepartsbiblioteker hentes gjennom pålitelige kilder som NPM og Maven.

- **A09-Security Logging and Monitororing Failures:** Hvis en applikasjon ikke inneholder et varslingssystem ved mistenkelig brukeroppførsler, kan dataangrep potensielt gå ubermerket. I denne applikasjonen finnes det ikke et slik system. Logger lagres på dedikert maskin, så det er mulig å følge med på og gå gjennom loggene.
- **A10-Server Side Request Forgery:** Ikke relevant i dette tilfellet, da det ikke gjøres eksterne kall basert på brukerinput.

Testing

Enhetstesting i Java har blitt gjort med JUnit, Mocking med Mockito og testrapportering med Jacoco. Som vist i Figur 30 har backend en total testdekning på 23%, med prioritering på testing av TBMMath-klassen. Frontend har ikke blitt programmatisk testet.



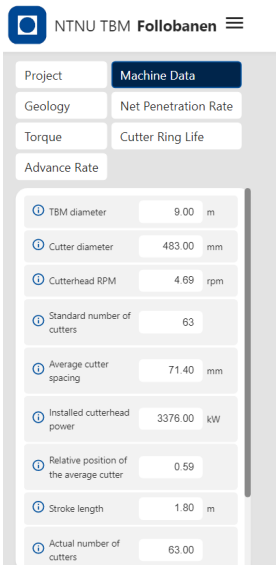
Figur 30: Rapport av testdekning

Nettleserstøtte

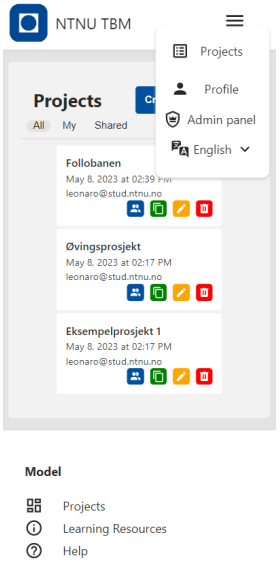
Nettsiden er støttet og testet på de mest brukte nettleserene som Edge, Safari, Firefox, Chrome og Brave.

Responsiv og rask nettside

Programmet har et responsivt design for skjermstørrelser helt ned til 300 piksler i bredden. Dette gjelder for alle sider, men til en mindre grad for administratorpanelet. Figur 31 viser hvordan designet ser ut på skjerm med mobilstørrelse. Kode på backend har blitt optimalisert ved å redusere antall databasekall fra server til database slik at spørringer skjer raskt.



(a) Brukergrensesnitt for dashboard



(b) Brukergrensesnitt for prosjekt. Elementer i navigasjonsmenyen kommer under en ikonknapp som har en dropdown-meny ved mobil-størrelse.

Figur 31: Brukergrensesnitter for mobil

Oversiktlig, konsistent og brukervennlig brukergrensesnitt
Det har blitt laget globale filer for darkmode og lightmode i CSS. Dette bidrar til konsekvent bruk av farger på hele nettsiden. Nettsiden har blitt gjort brukervennlig gjennom oppfylling av kravene i henhold til WCAG 2.1. Brukergrensesnittet har blitt gjort oversiktlig gjennom bruk av lister for visualisering av store mengder data og at dashbordet er designet som en tabell.

Skalerbarhet
Systemet har blitt gjort skalerbart gjennom å følge konvensjoner for programmering i Vue og Spring Boot og normalisering av databasemodell. Programvaredesignmønster CSR har blitt brukt på backend for oversiktelig plassering og samhandling mellom klasser. Systemdokumentasjon som JavaDoc, kommentarer i kode, dokumentasjon av endepunkter gjennom Swagger, Vedlegg E: Systemdokumentasjon som databasemodell og Vedlegg D: Kravdokumentasjon gjør det enklere for andre utviklere å sette seg inn i det eksisterende systemet og jobbe videre på det.

4.3 Administrative Resultater

4.3.1 Framdriftplan

Under arbeidet med Vedlegg A: Forprosjektplan ble det definert en plan for tidsforbruk gjennom et GANTT-diagram. Diagrammet viser start og stopp for aktivitet. Framdriftsplanen har stemt godt med unntak av sprint 5 som har blitt fjernet, sprint 4 har blitt forlenget og prototyping har vært en kontinuerlig prosess gjennom prosjektet i motsetning til en sammensatt blokk. Diagrammet har vært nyttig for gruppen for å vite start og stopp for kategori, men sier lite om tidsforbruket.

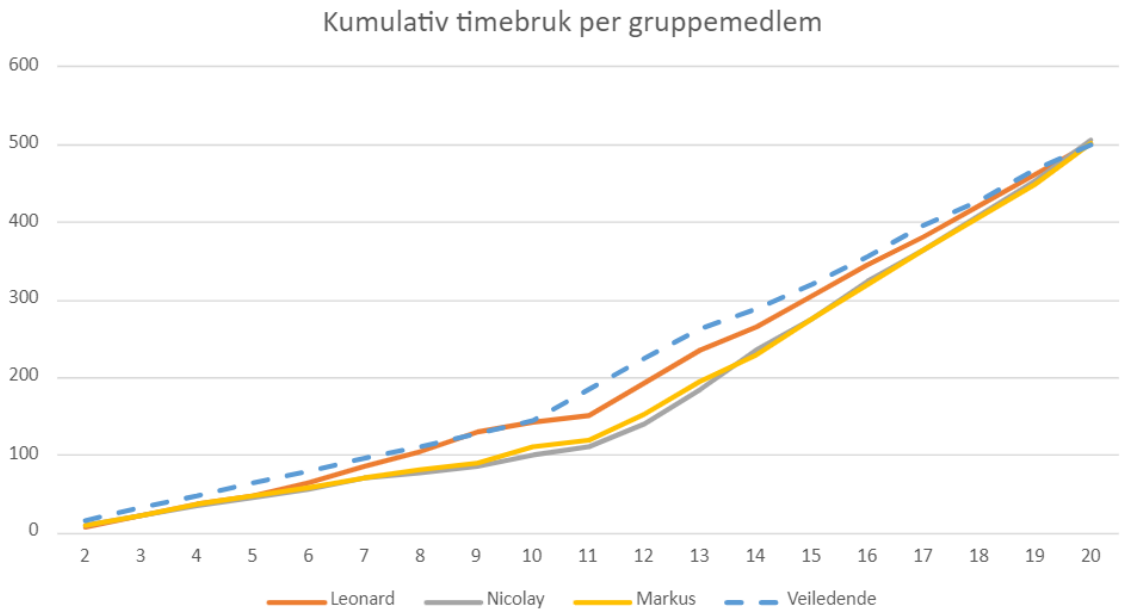
4.3.2 Timefordeling

Timer har blitt skrevet for hver arbeidsdag i Vedlegg B: Prosjekthåndbok og har blitt koblet opp mot kategori i GANTT-diagrammet fra forprosjektplan. Flere kategorier som møter, administrativt, undervisning og systemdokumentasjon har blitt lagt til for å vise et mer detaljert tidsforbruk.

Forprosjektplan og poster inngår i obligatorisk arbeid. Visjonsdokument og kravdokumentasjon inngår i kategorien planlegging. Krav for timeforbruk for hvert medlem var angitt til 500 timer, som har blitt oppnådd. Timeforbruk opp mot veiledende timeforbruk har vært nok så god ifølge Figur 32.

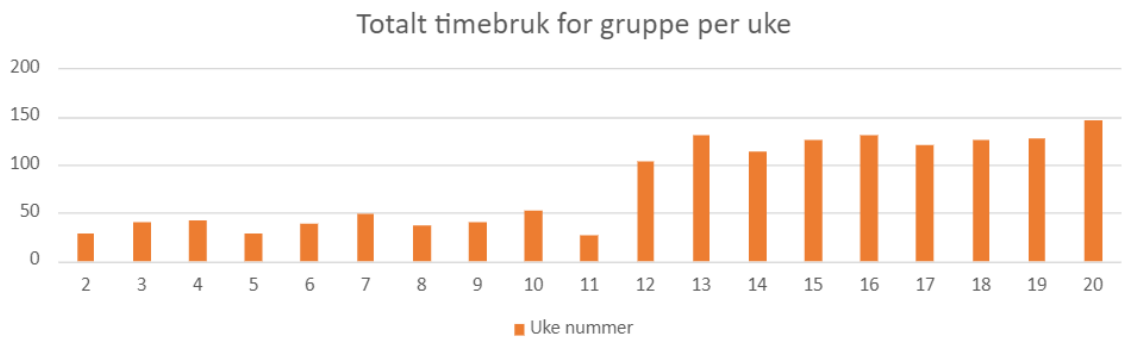
Tabell 2: Tidsforbruk

Kategori	**Tidsforbruk**
Sprint 1	206,5
Sprint 2	237
Sprint 3	156
Sprint 4	268,5
Hovedrapport	352,5
Prototyping	17
Administrativt	6,5
Møter	43
Obligatorisk arbeid	59,5
Undervisning	31,5
Planlegging	96
Systemdokumentasjon	35,5
Totalt	1509



Figur 32: Kumulativ timebruk per gruppemedlem

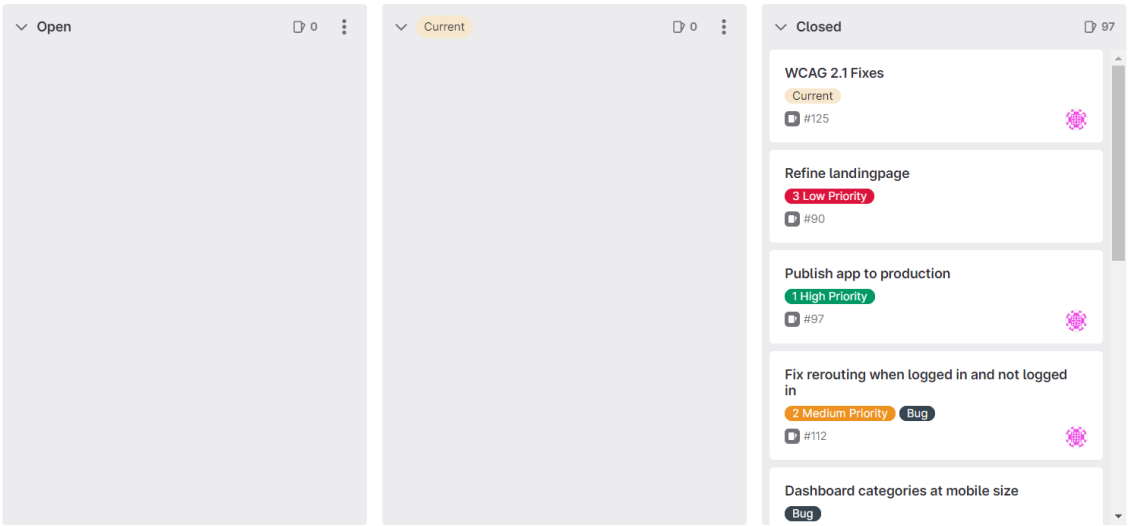
Figur 32 viser hvor mange kumulative timer hvert gruppemedlem har underveis i prosjektet. Veilende viser hvor mange kumulative timer som var veilendende gjennom prosjektet. Veiledende var laget i forbindelse med forprosjektplanen og har blitt brukt for å holde hvert gruppemedlem i rute. Figur 33 viser at totalt timebruk for gruppe har vært jevnt fra uke 2 til 11 og fra uke 12 til 20. Tabell 2 viser fordelingen mellom de forskjellige aktivitetene. Utvikling står for størst antall timer etterfulgt av hovedrapport.



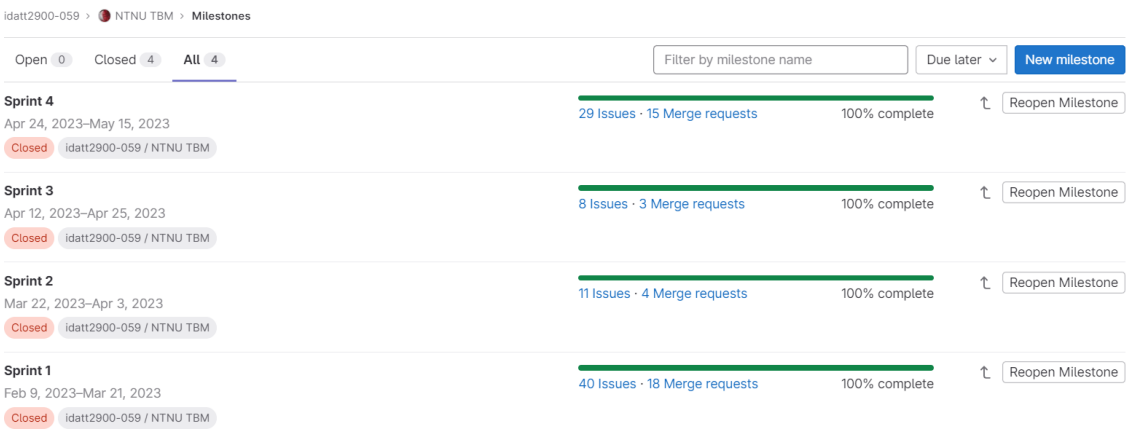
Figur 33: Totalt timebruk for gruppe per uke

4.3.3 Utviklingsmetodikk

Gruppen har benyttet seg av smidig utviklingsmetodikk scrum. Produktkø ble opprettet før hver sprint med rangering basert på prioritet gitt fra produkteier. For hver standup har gruppemedlemmene fortalt hva det har gjort, hva det skal jobbe med og utfordringer knyttet til dette. Etter endt sprint har gruppen vist fram resultatet til produkteiere. Resultatene av sprint review kan finnes i Vedlegg H: Sprint Review. Videre har gruppen brukt GitLab Issueboard for å holde en oversikt over produktkøen. Figur 34 viser at hvert issue blir plassert i en av kollonnene: Open, current og closed. Open er for issues som er klar for å begynnes på. Current er issues som for øyeblikket blir jobbet på og closed er issues som er ferdig. For at et en issue skal anses som ferdig, må pull requesten for denne issuen ha vært gjennom review av de andre gruppemedlemmene. For hvert issue knyttes det opp tags som for eksempel 'bug', 'prioritet', 'frontend' og 'backend'. Videre har hvert issue et gruppemedlem knyttet opp mot seg. Issuene inngår i en milestone (Sprint), Figur 35, som har et start- og sluttidspunkt. Vedlegg G: Sprint planning viser en oversikt over sprint backlog for hver sprint med prioriteringer. Det er ingen backlog for sprint 1, siden sprint 1 hadde et enkelt mål: Implementere modellen. Dette innebærer implementasjon av matematiske utrykk programmatisk på backend. Vedlegg I: Backlog viser en sammenstilling av alle issuene på GitLab koblet opp mot sprinter (milestones).



Figur 34: GitLab Issueboard etter endt utvikling, 19.05.2023



Figur 35: GitLab Milestones (Sprinter) etter endt utvikling, 19.05.2023

5 Diskusjon

Diskusjonsdelen tar for seg drøfting av resultatene i kapittel 4 opp mot problemstillingen. Kapittelet inneholder drøfting om hva som gikk bra og hva som kunne blitt gjort bedre.

5.1 Diskusjon Av Vitenskaplige Resultater

Gruppens digitalisering av NTNU-modellen for TBM tilbyr mer detaljerte formler enn det som er beskrevet i den originale modellen. Gjennom samarbeid med oppgavestillerne har nyutviklede formler blitt brukt istedenfor de originale fra modellen. Dette var ikke en del av oppgaveteksten, men siden oppgavestillerne har pågående arbeid innenfor videreutvikling av modellen var dette lett tilgjengelig materiale som gruppen kunne inkludere i produktet.

Under oversettingen fra oppgavestillernes Excel-ark til gruppens Java-kode var det nødvendig å endre på strukturen til logikken, ettersom de to teknologiene er forskjellige. Det var under denne oversettingen at muligheten for interpolering ble identifisert og foreslått fra gruppen. Under veiledning fra oppgavestillerne implementerte gruppen disse interpoleringene, og bidro til videreutvikling selv. Dette var også funksjonalitet som ikke ble spesifisert i oppgaveteksten, men en mulighet som dukket opp under det tverrfaglige samarbeidet, og en forbedring som sammenfaller med det ingeniørfaglige målet om å digitalisere modellen.

5.2 Diskusjon Av Ingeniørfaglig Resultater

Gruppen har fullført de fleste mål definert i visjonsdokumentet. Noen mål har blitt nedprioritert og dermed ikke fullført på grunn av manglende tid. Punkter dette gjelder for er utskrift av prosjekt i PDF-format, endring av verdier i sanntid (WebSocket) og usikkerhetsberegninger i modellen. Samtidig har funksjonelle krav blitt endret, og det er funksjonalitet som har blitt nevnt på de første møtene som ikke lenger var relevant å implementere på slutten av prosjektet, for eksempel input-verdier til parameter ved hjelp av dropdown.

5.2.1 NTNU-Modellen for TBM

Modellering av soner var et uforutsett hinder som behøvde en ekstra Sprint for å fullføre. Ved første øyekast kan oppdelingen av et fjell i mindre soner se ut som en liten del av NTNU-modellen, mens i realiteten er mesteparten av utregningene på en per-sone basis og resultatverdiene for prosjektet er ofte bare en sum av sone-resultatene. Gruppens initielle utregningsarkitektur baserte seg på antagelsen at sone-logikk var en liten tilleggsmodul til et mer prosjektrettet system, og det ble derfor en utfordring å implementere soner.

Ett av de funksjonelle kravene fra oppgavestillerne var å inkludere usikkerhet i beregningene for å ta hensyn til at ikke alle forhold er fullstendig kjent i et tunnelprosjekt. Idéen var å legge til en sannsynlighetsfordeling til relevante parametere, der brukeren kunne i en egen fane be tjeneren beregne resultat hundre- eller tusenvis av ganger for å se utfallsrommet til viktige resultat som ukentlig fremdrift eller kutterlevetid. Som vist i prioriteringslistene fra Sprint 2, 3 og 4 i Vedlegg G har modellering av usikkerhet vært av høy prioritet, men aldri første prioritet. Ved hvert planleggingsmøte ble begge parter enige om at en ferdigstilt og grundig testet modell var viktigere enn å legge til denne funksjonaliteten.

Et viktig krav fra produkteier var at dashbordet skulle ha et 'tabell-format'. Dette var på grunn av at det ville være mer oversiktlig og stå i stil med oppsettet for parametere i Prosjektrapporten HARD ROCK TUNNEL BORING [5], hvor parametere kommer i riktig rekkefølge i henhold til utregning. Dette viste seg å være utfordrende spesielt med tanke på WCAG-kravene for endring av skriftstørrelser. Dette var forsøkt løst, men fjernet siden dette ville medføre drastiske endringer på dashbordet. Siden produkteiere var fornøyd med designet, veide dette opp for kravet, men ved videre utvikling er dette et punkt som bør forbedres.

Kategorier var en nødvendig abstraksjon på grunn av at modellen inneholder mange parametere og det kan fort bli uoversiktlig om alle parameterene vises på samme side. Derfor viser dashbordet parametere basert på kategorien. Dette har gjort dashbordet mer oversiktlig og parametere som ofte inngår i samme utregning kommer sammen. I tillegg til kategori var det også et ønske fra produkteier at parametere skulle komme i samme rekkefølge som utregningene. Dette viste seg å være en utfordring på grunn av at kravet kom sent i utviklingsprosessen, i sprint 4. Siden det var begrensende tid igjen var det fordelaktig å implementere dette som en frontend-funksjonalitet istedet for på backend, da en backend-implementasjon ville medføre mer programmering. For å lage denne funksjonalitet mer skalerbar, bør det lages et system for rekkefølge på parameter.

Et interaktivt boksplott for valg av DRI og CLI var et ønske fra produkteiere, slik at det ikke var nødvendig å undersøke doktoravhandlingen [1] for å vite hvilke verdier som er gyldige. Boksplottet er utformet ved hjelp av CSS-Grid hvor hver rad tilsvarer 100 ruter. Rutene blir deretter fylt med forskjellige streker ut i fra karakteristikken til steintypen. Systemet er designet på en måte som gjør det skalerbart. For hvert parameter kan det kobles opp steintyper. Når et parameter har steintyper, kan frontend dynamisk vise redigeringknapp. Det er dermed inneholdet i databasen som avgjør om denne typen redigering skal være mulig. For vinkelkalkulator må det spesifiserer hvilke parameter som skal kunne hente verdi fra denne på frontend. Ideelt sett skulle dette også blitt styrt fra databasen, men på grunn av at det kun er et parameter som støtter denne funksjonaliteten, var det denne løsningen som ble valgt. Implementasjon av boksplott og vinkel-kalkulator støtter opp mot kravet av at det skal være tilgjengelige flere muligheter for input der det ses nødvendig i henhold til NTNU-Modellen for TBM [1].

5.2.2 Brukersystem

Brukersystemet er, bak selve modellen, det nest viktigste aspektet ved produktet. Dette er fordi all annen funksjonalitet bygger på det. Et godt eksempel på dette er at brukere gjør at man kan dele på prosjekter, og dermed samarbeide på TBM prosjekter på en måte som ikke var mulig tidligere. Brukersystemet tilfredstiller de funksjonelle kravene som ble gitt på en god måte. Funksjonalitet som å lage en bruker, logge inn og ut og annen funksjonalitet knyttet til dette som en landingsside, navigering, glemt passord, sletting av bruker osv., har blitt fullført.

En viktig årsak til at brukersystemet ble komplett var knyttet til viktigheten av det. Hadde ikke brukersystemet påbegynt tidlig nok ville det vært stort potensiale for at det ble en flaskehals. Det ble derfor høyt prioritert å starte tidlig og opprettholde høy kvalitet, noe som synes i resultatet. En annen årsak til at brukersystemet ble velykket var teknologiene som ble valgt. Spesielt bruken av Spring Security sørget for å holde systemet av høy kvalitet og redusere feilen for menneskelig utviklerfeil i kritisk funksjonalitet som passordhåndtering. Den siste årsaken til at systemet fungerer godt er nok knyttet til planleggingsprosessen. Før det ble implementert ble det gjort grundig planlegging av entiteter, passordlagring, roller og tokensystem. Utviklingsprosessen ble dermed raskere, og trolig også av høyere kvalitet.

5.2.3 Prosjektsystem

Prosjektsystemet har blitt implementert med all den funksjonalitet produkteierne har definert som viktig. Prosjektsystemet er en sentral del av det som skiller denne løsningen fra eksisterende løsninger, fordi det tilrettelegger for en ny type samhandling med modellen. Det har derfor vært av høy prioritet å få dette inn på en god måte. I begynnelsen definerte også produkteierene funksjonalitet for prosjektsystemet som ikke har blitt laget. Dette inkluderer eksport av et prosjekt til en formatert PDF, og live redigering med andre medlemmer av prosjektet. Årsaken til at dette ikke ble implementert var at det ble nedprioritert fordi det var å anse som uviktig. For eksempel ville det ha ingen nytte å kunne eksportere et prosjekt til PDF dersom den matematiske modellen ikke ble ferdigstilt. Således er dette et svært godt eksempel på at prioriteringsprosessen som ble brukt var vellykket.

5.2.4 Administratorpanel

Et viktig element av oppgaven var å gi produkteierne gode administratorverktøy, siden de ikke har utviklere tilgjengelige etter prosjektets slutt. Administratorpanelet fungerer hovedsaklig som en vedlikeholdsplatform, men også en mulighet for å rette opp i eventuelle feil gruppen har gjort som blir oppdaget i etterkant.

Det første verktøyet er en fane for parameter-administrasjon vist i Figur 25. Denne gjør det mulig for produkteierne å endre grenser, standardverdier, enheter og antall desimaler for parametere. I fanen for språk vist i Figur 26 kan alle tekster brukt i brukergrensesnittet endres på, både for norsk og engelsk versjon. Det ble kommunisert fra oppgavestillerne at fagord kunne endre seg som tiden går, slik at oppdatering av tekst var viktig. Det var også viktig med en norsk versjon av nettsiden siden det var optimalt for undervisningssammenhenger. I tillegg er det visse tekstfelter der det er mer naturlig at oppgavestillere skriver selv, slik som tunnelfaglige tekster og brukervilkårene en ny bruker må godta ved registrering. Det siste administratorverktøyet, vist i Figur 27, gir administratorbrukere mulighet til å gi andre administratorrettigheter. Denne funksjonaliteten sørger for at produktet kan fortsette i drift selv om de originale produkteierne bytter stilling.

5.2.5 Språkstøtte

Språksystemet til produktet er en egenimplementert løsning, det tar ikke i bruk det populære rammeverket i18n [21]. Det initielle valget om å designe et eget system ble tatt på grunn av usikkerhet rundt hvordan rammeverket håndterer språkfiler fra en database istedenfor det klassiske oppsettet med en statisk språkfil lagret i filstrukturen til frontend. I retrospekt ser det ut som språksystemet vårt kunne tatt i bruk i18n, men det egne systemet gruppen brukte fungerte meget bra. Språksystemet er beskrevet i detalj i Vedlegg E: Systemdokumentasjon.

En utfordring med arbeid med språksystemet har vært at all tekst står i teksttabellen i databasen, mens det som normalt er tekstfelter i andre tabeller inneholder referanser til språktabellen, noe som har gjort lesing av databasen i forbindelse med feilsøking og utvikling mer omfattende.

5.2.6 Epost

At brukere må registrere eposten sin var viktig for oppdragsgiver. Dette fungerer blant annet som et tiltak for å øke terskelen for å få bruke tjenesten. I tillegg er de andre epostjenestene som glemt passord og prosjektinvitasjon avhengig av at brukeren faktisk har tilgang til epostadressen sin.

På tross av at epost egentlig er en del av bruker- og prosjektsystemet er det satt som et eget punkt. Dette er en del av den iterative prosessen som ble brukt. De andre systemene fungerer uten epost, og da får man de oppe og går raskere. Dette gjorde at disse elementene kunne bli testet og bygget videre på tidligere i prosessen.

5.2.7 Ikke-Funksjonelle Egenskaper

WCAG
Nettsiden klarer å oppfylle de fleste av kravene som blir spesifisert i WCAG 2.1 blant annet riktig bruk av tagger, dynamisk tilpasning, språk, kontrastnivåer og tastaturstøtte. Detaljert informasjon om dette kan finnes i Vedlegg F: WCAG-sjekkliste. Dette er et godt resultat men det er mangler et punkt for å få full uttelling. Nettsiden består stort sett kravet 1.4.4: Endring av tekststørrelse, bortsett fra dashbordet. Her har mange av tekstene en definert størrelse på 12 piksler, noe som ikke gjør dem overstyrbar med nettleseringinstillinger for tekststørrelse. Dette har blitt gjort for bedre plassutnyttelse gjennom ønsker fra produkteier om et 'Tabell-format' på dashbordet. Dette gjør informasjon oversiktlig og komprimert, på bekostning av universell utforming.

Sikkerhet
Produktet som lages er ikke bare et lokalt utviklingsprosjekt, men et produkt som faktisk skal ut på nettet. Dette er bakgrunn for at sikkerhet er spesielt viktig i denne applikasjonen.

Hashing av passord er er et kritisk tiltak for å beskytte sensitiv data for brukeren. Algoritmen som brukes, bcrypt, ligger på listen til OWASP over anbefalte hashingalgoritmer [22]. Styrkejusteringen er basert på den abefalte tiden på 250-1000ms, som er en avveining mellom brukeropplevelse og sikkerhet. Videre er tokensystemet er nødvendig for at brukeren ikke skal måtte oppgi passordet sitt ved hver forespørsel, samtidig som at en ondsinnet aktør ikke skal kunne utgi seg for å være noen andre.

Gjennomgangen i av OWASP Top 10 i Kapitell 4.1.7 viser at applikasjonen er godt beskyttet mot vanlige sikkerhetshull. Likevel vil det være naivt å definere applikasjonen som sikker. Det finnes svært mange angrepsmuligheter for ondsinnede aktører. I tillegg utvikles stadig nye og forbedrede former for angrep, og siden applikasjonen ikke blir aktivt vedlikeholdt er den spesielt utsatt for dette. I sum anser vi applikasjonen som tilstrekkelig beskyttet basert på OWASP Top 10 og ønskene til produkteier.

En viktig årsak til at denne delen ble vellykket er knyttet til prosess og teknologivalg. Sikkerhetsaspektet ved applikasjonen har vært et tema helt fra starten og det ble gjort grundig research og planlegging. I tillegg har teknologivalg som Spring Security, som har mye innebygd funksjonalitet og god dokumentasjon, ført til et mer pålitelig resultat.

Testing
Det har ikke blitt gjort enhetstesting på frontend. Den første grunnen er tidsbegrensinger. For at gruppen skulle klare å fullføre størst antall av de funksjonelle kravene og kompleksiteten av kravene gitt av produkteier, som for eksempel implementering av modellen innen tidsfrist, bestemte gruppen å ikke enhetsteste frontend. Videre har det vært kontinuerlig endring i funksjonalitet og design som har gjort det lite hensiktsmessig å implementere enhetstester, for å så måtte oppdatere de etterhvert disse endringene har blitt gjort. Dette har medført til mer tid til implementering av funksjonalitet og design. Selv om det ikke har blitt gjort enhetstesting på frontend, har dette blitt veiet opp med et større fokus på manuell testing fra utviklerene og produkteieren på den publiserte nettsiden. Om tidsbegrensingene hadde blitt fjernet og at produktet konvergere mot et mer stabilt produkt med mindre store endringer, ville argumentet for å enhetsteste frontend vært større.

Backend har enhetstester og har en total testdekning på 23%. Testdekningen skulle optimalt vært høyere men av samme grunn som manglende testing av frontend ble dette bortprioritert. Hovedfokuset på enhetstesting har vært av klassen TBMMath, hvor testdekningen er på 84%. Klassen er sentral i applikasjonen siden denne holder på alle matematiske funksjoner, og er derfor svært viktig å teste. Grunnen til at dekningen ikke er på 100% er siden resterende linjer er multiplikasjon, divisjon, eller lignende operasjoner som man kan stole på at Java gjennomfører korrekt.

Nettleserstøtte
Gruppemedlemmene og produkteier har testet applikasjonen i flere nettlesere både på PC og mobile-enheter. Gruppen støtte ikke på noen problematikk knyttet til bruk i flere nettlesere. Ved å fokusere og prioritere det mest brukte nettleseren har gruppen kunne dekke behovet til mesteparten av markedet. Nettleseren dette gjelder er Chrome, Safari, Edge og Firefox som står for 91,7% av markedsandel i 2022 [23].

Responsiv og rask nettside
Et krav fra produkteiere var at nettsiden skulle ha et responsivt design for bruk på mobile enheter som telefon og pad. Måten dette har blitt implementert på er med bruk av Grid og Media Queries med breakpoints i CSS. Administratorpanelet er også til en viss grad responsivt, men dette var ikke et krav fra produkteiere fordi panelet kun blir brukt av et fåtall personer og bruken av det blir prioritert til PC.
Et problem i slutten av utviklingsfasen var at nettsiden var treg ved oppdatering av modellen. Problemet på backend var at det var for mange unødvendige kall til databasen. Dette ble løst ved

å samle lese- og skrivekallene til én spørring. Et parameter som ble stort påvirket av optimalisering var TBM-diameter hvor det kunne ta opp til 50 sekunder å fullføre forspørselen, og etter endring endte dette på under 1 sekund.

Oversiktlig, konsistent og brukervennlig brukergrensesnitt
Brukergrensesnittet har optimalisert gjennom bruk av flere wireframes, prototyper og tilbakemelding fra produkteier. En av de store utfordringer var å vise mye data på en organisert og naturlig måte, i henhold til relasjonen mellom parameterene. Det er betydelige forandring som må ha blitt gjort, spesielt på dashbordet, basert på tilbakemeldinger fra produkteier. Produkteiere ville ha en tabell-aktig oversikt over parameterene, som samtidig skulle være kategorisert og komme i riktig rekkefølge basert på utregningsrekkefølge i modellen. Videre har det vært flittig bruk av 'luft' i applikasjonen, som skaper et mer tydelig skille mellom hvilke komponenter som henger ihop og hvilke som ikke gjør det. Ved bruk av globale CSS filer har gruppen kunnet implementere darkmode, samtidig som det blir konsistent bruk av farger. Dette har fungert bra, spesielt programmeringsmessig, hvor det er enkelt å style nettsiden kun ved bruk av et fåtall globale variabler. Ved å følge WCAG 2.1 har det ikke-funksjonelle kravet om oversiktlig, konsistent og brukervennlig brukergrensesnitt blitt oppfylt, siden kravene spesifisert i WCAG anngår nettopp dette. Selv om det ikke-funksjonelle stort sett har blir gjort på en god måte, ville en forbedring vært å gjort applikasjonen mer brukervennlig for de med funksjonshemmering. Mangler her blir også spesifisert i delen om WCAG, og med mer tid kan kravene oppfylles.

Skalerbarhet
Skalerbarhet har vært i fokus under hele utviklingsprosessen. Systemet er dokumentert gjennom JavaDoc i backend, kodekommentarer, Swagger for endepunkt, og systemdokumentasjon. Dokumentasjonen er forhåpentligvis dekkende nok for at en ny utvikler kan sette seg inn i systemet. Gjennom Sprint 3 og 4 ble systemet mer og mer komplekst, spesielt på grunn av sone-implementasjon som skapte en stor refakturering av koden både i front- og backend. Som prosjektet nærmet seg sluttdato ble et ferdigstilt produkt prioritert over den strukturen som var ønskelig. Likevel mener vi at bruken av arkitekturmønster og konvensjoner for programmering i Vue og Spring Boot har gjort systemet skalerbart nok til å kunne utvides på en god måte i fremtiden.

5.3 Diskusjon Av Administrative Resultater

5.3.1 Framdriftplan

GANTT-diagramemt definert i Vedlegg A: Forprosjektplan har vært et godt utgangspunkt for gruppen i å planlegge forskjellige aktiviteter samt start- og sluttidspunkt. Likevel har GANTT-diagrammet mangler. Diagrammet består av en sammensatt planleggingsblokk, som i realiteten har vært en kontinuerlig prosess. Sprint Review har for eksempel vært en del av planleggingen, og denne har foregått regelmessig gjennom utviklingsfasen. Grunnet dynamisk endring av funksjonalitet og tilbakemeldinger fra produkteiere, har det på flere tidspunkter vært nødvendig med møter som ellers ikke har inngått i denne planleggings-blokken. Oppstart, forprosjektplan, visjonsdokument og kravdokumentasjon stemmer godt med aktivitetstidspunkt. Utviklingfasen stemmer også bra, med unntak av sprint 5 som har blitt fjernet til fordel for en lengre sprint 4. Det var ikke hensiktsmessig for gruppen å planlegge en ny sprint på den korte tiden som var igjen. Ordinært sett var sprint 5 ment for ferdigstilling av koden. Dette ville betydd at gruppen måtte ha arrangert sprint review for en sprint som i realiteten ikke var ment for ny funksjonalitet, og prioritering av funksjonalitet som dette medfører. Ifølge GANTT-diagrammet ble det også gjort en estimering at sprint 1 måtte være litt over dobbel lengde som de andre sprintene, på grunn av tidsbegrensninger med tanke på et annet fag som gikk i parallelt med bacheloroppgaven. For at sprintene skulle ende opp med cirka samme antall timer, måtte denne sprinten gå over en lenger periode. Til sammenligning med faktisk antall timer brukt i Tabell 2, stemmer dette estimatet nok så godt. Sprint 1 har cirka samme antall timer brukt som de andre sprintene.

5.3.2 Timefordeling

Timefordeling ble originalt gjort basert på GANTT-diagrammet. Dette viste seg å ikke være detaljert nok for å få et riktig bilde av timefordelingen. For eksempel endte prosjektet med å ha en del timer fordelt på administrativt, møter, undervisning og systemdokumentasjon. Antall timer tilsvarer cirka 120 timer. Derfor har det i ettertid blitt lagt til flere kategorier. Et mål for gruppen var at sprintene skulle ha rundt samme tidsforbruk. Dette var ment som et argument for å enklere sammenligne resultater for hver sprint opp mot tidsforbruket. Dette målet ser ut til å ha blitt oppnådd nok så godt, hvor hver sprint har cirka 200 timer +- 50 timer i følge Tabell 2.

Gruppen har koblet timelista opp mot et veiledende diagram, se Figur 32, for antall forventede timer per uke per gruppemedlem. Dette diagrammet har vært en god måte å vite om gruppemedlemmene var i rute på antall forventede timer brukt per uke. Basert på diagrammet kan man se en tydelig økning på antall timer per gruppemedlem etter uke 11. Grunnen til denne økning, og også et estimat av økning, var på grunn av at emnet INGT2300 som gikk i parallel med bacheloroppgaven, var fullført. Etter dette kunne gruppen bruke hele uken på bacheloroppgaven. Totalt antall estimert timer har blitt oppnådd av hvert gruppemedlem og gruppen total sett. Både timeforbruk før og etter eksamen har omtrent vært lineært, noe som vil si at timeforbruket har vært forutsigbart. Dette ser vi på som positivt.

Resultatet av antall timer for utviklingen er et forventet resultat med tanke på at oppgaven er utviklingsorientert. Timeforbruk for utvikling ligger på 868 timer, som tilsvarer 58% av tidsforbruket. Siden det har vært tett samarbeid med produkteier, har det også gått en del timer til møter. Dette inkluderer også sprint review, som har blitt fullført etter hver sprint.

5.3.3 Utviklingsmetodikk

Scrum har fungert bra for iterativ utvikling av produktet. I ettertid kan det argumenteres at denne typen utviklingsmetodikk var for omfattende basert på antall gruppemedlemmer og relasjonene mellom gruppemedlemmene og at et enkelt kanban-board hadde vært nokk. Scrum artefakten retrospective har for eksempel ikke blitt utført, fordi gruppen har jobbet med scrum tidligere, og har sånn sett 'optimalisert' den iterative prosessen. Argumentet var at denne artefakten ikke var en god prioritering av tid, og at denne tiden heller kunne bli brukt til utvikling, siden resultatet av scrum retrospective stort sett ville vært det samme: God kommunikasjon mellom gruppemedlemmer, effektiv koordinering ved bruk av backlog og klar og tydelig planlegging for hver sprint. Hadde gruppen vært større ville argumentet vært sterkere for bruk av scrum retrospective. Gruppen har heller ikke tatt i bruk scrum master på grunn av størrelsen på gruppen. Det har likevel blitt brukt en gruppeleder til møter som scrum review.

Før starten av hver sprint har en GitLab issueboard milestone blitt fylt opp med sprint backlog, rangert etter prioritering. Dette har fungert godt og har vært nyttig for å ha en god oversikt over arbeidsoppgavene til alle gruppemedlemmer, og for å vite hva som er nyttig å jobbe med videre etter endt arbeidsoppgave. Det er ikke noen forbedringspotensialer å nevne med bruken av issueboardet, siden dette har fungert bra.

Utviklingsmetodikken som har blitt brukt har fungert bra. Siden scrum som har blitt brukt i dette prosjektet har blitt modifisert til en viss grad (Ingen retrospective og scrum master), kan man argumentere for at det burde ha blitt gjort en grundigere vurdering av utviklingsmetodikk i startfasen. En mer passende utviklingsmetodikk kunne for eksempel ha vært Kanban. Likevel har standup og scrum review med utfylling av backlog vært veldig nyttig for gruppen.

5.4 GruppereflekSjon

Siden gruppen har jobbet sammen tidligere med stor suksess har samarbeidet og kommunikasjonen i gruppen vært god. Gruppen har stort sett arbeidet sammen på grupperom og tatt lunsj sammen. Dette har vært bra for moralen, det sosiale aspektet og for effektiv kommunikasjon. Kunnskapsnivået i gruppen har vært tilstrekkelig til å utvikle et produkt som produkteier kan ta godt nytte

av og som gruppen er stolt av å presentere. Videre har gruppemedlemmene utfylt hverandre med at noen fortrekker å jobbe på backend, imens andre fortrekker frontend. Det har også vært forskjeller i ønsket om å utvikle, skrive hovedrapport og drive med administrativt arbeid som organisering av dokumenter, møteinnkallelser, møtereferater og være møteleder. Disse ønskene har også blitt oppfylt på en god måte siden gruppemedlemmene stiller med forskjellige preferanser. Gruppen er fornøyd med jevnt og forutsigbart timeforbruk gjennom prosjektet. Dette har vært til god nytte for å unngå skippertak, noe som ofte kan dukke opp i avslutningsfasen av prosjektoppgaver. Videre har gruppen brukt kommunikasjonskanaler som Messenger flittig til å stille spørsmål og planlegge. Dette har vært veldig nyttig for gruppen til å være effektiv og raskt tilpasse seg ny informasjon, som for eksempel eposter fra produkteiere. En nedside for gruppen mot slutten av prosjektet har vært presset om å kunne levere et omfattende og komplekst system innen kort tid. Dette har ført til dårligere kvalitet av PR reviews enn det gruppen har ønsket. Mot slutten av prosjektet har PR reviews hatt større fokus på manuell testing istedet for å gi tilbakemelding på om kodekvalitet er grundig, optimalisert og velimplementert. Dette er en prioritering som gruppen har måttet tatt på grunn av manglende tid og for at kritisk funksjonalitet skal ha blitt raskere implementert.

Om gruppen skulle ha gjort prosjektet på nytt ville vi tatt i bruk sprint retrospective. Selv om gruppen er kjent med hverandre, jobbet med scrum før og under tidsbegrensninger, er det fortsatt nyttig å evaluere kvaliteten av hver sprint. På den måten sikrer vi at kvaliteten på sprinten optimaliseres. Et annet punkt er å slutte programmeringen på et tidligere tidspunkt, slik at kvaliteten på PR reviews forblir av høy kvalitet.

6 Konklusjon Og Videre Arbeid

Konklusjonsdelen presenterer konklusjonen som kan trekke i forhold til problemstillingen. Konklusjonen følger av resultatene og diskusjonen. Til slutt gis det forslag og anbefalinger til videre arbeid.

6.1 Konklusjon

Oppgaveteksten stiller krav til å utvikle et brukervennlig og nettbasert verktøy som baseres på Javier Macias sin phd-avhandling [1]. En forståelse av systemet som er utviklet krever grunnleggende teori om hva tunnelboremaskiner er, om NTNU-modellen, modell-parametere, geologiske soner og fraktursett. Videre kreves en forståelse av CSR-arkitektur, smidig utviklingsmetodikk og de tidligere løsningene som eksisterer.

Gruppen har valgte teknologier passende til problemstillinger og utviklingsmetodikk som sikrer god kommunikasjonsflyt med produkteier og god planlegging. Prototyping og wireframing har blitt brukt for å planlegge, scrum som smidig utviklingsmetodikk muliggjort ved hjelp av GitLab og arbeids- og rollefordelingen har vært fleksibel.

Sluttproduktet kan anses som et komplett system med brukersystem, prosjektsystem, e-post, språkstøtte, administratorpanel og implementasjon av NTNU-modellen for TBM. De funksjonelle kravene spesifisert i visjonsdokumentet har blitt fulgt og stort sett bestått. Implementasjonen av Javier Macias sin phd-avhandling står som fundamentet i applikasjonen og består av 81 parametere med 49 tilhørende matematiske funksjoner. Systemet gir tilbakemelding på feil inntastinger og har støtte for spørsmål og opplæring om TBM. For å oppfylle kravene om at applikasjonen både skulle være brukervennlig og sikker har både WCAG og OWASP blitt fulgt, hvor kravene stort sett har blitt bestått.

Administrativt har gruppen brukt et GANTT-diagram som framdriftsplan som har fungert godt til estimering av start- og sluttidspunkt for aktivitet, men det finnes manglende detaljnivå som underveis har blitt måttet lagt til. Utvikling av produktet har stått som en stor del av timefordelingen og dette var forventet basert på at oppgaven har stort utviklingsfokus. Scrum har blitt brukt som utviklingsmetodikk men det kan argumenteres om denne metodikken er passende for en gruppe på denne størrelsen. Likevel har Scrum vært nyttig på grunn av artefaktene som scrum review, backlog og standup. Scrum har igjen blitt gjort mulig av GitLab issueboard som har fungert godt til organisering av arbeidet.

6.2 Videre Arbeid

For personer som eventuelt skal jobbe videre med vår applikasjon eller som skal jobbe på en lignende problemstilling anbefaler vi et grundig forarbeid. I et stort system som NTNU-modellen er det kritisk å kjenne kildemateriellet grundig, og å skaffe oversikt over utfallsrommet til modellen. Gruppen har hatt god erfaring med en iterativ utviklingsmetodikk i samarbeid med oppgavestiller, og er en metodikk som anbefales videre i et tverrfaglig prosjekt.

Grunnet tidsbegrensninger har ikke all ønsket funksjonalitet blitt implementert. Beregning av usikkerhet gjennom Monte Carlo Simulering er en viktig funksjonalitet som er med på å styrke resultatene i modellen og står som førsteprioritet ved videre utvikling.
Videre kan det implementeres visualisering av data i form av tre-struktur eller grafer, siden dette kan være spesielt nyttig i undervisningssammenheng. På den måten kan sammenhengen mellom de forskjellige parameterene i NTNU-modellen bli forstått på en bedre måte.
Utskrift i form av PDF-format ble prioritert bort tidlig i utviklingsfasen, men kan fortsatt være en nyttig funksjonalitet å utvikle. Det er ikke alltid man har tilgang til PC eller mobil og da kan for eksempel prosjektet printes ut og fysisk kopi kan bli brukt ute ved et TBM-prosjekt eller i andre sammenhenger.
Til slutt er kostnadsberegninger et nyttig verktøy, spesielt for sammenligning mellom kostnader på

TBM-prosjekter og prosjekter med sprengning. Det er også nyttig ved optimalisering av kostnader, og for å få en oversikt over hvor om man har mulighet til kutte kostnader. Denne funksjonaliteten ble bortprioritert tidlig i utviklingsfasen til fordel for implementasjon av selve modellen. Ved implementasjon av ny funksjonalitet anbefales det å gjøre et grundig litteratursøk og sette seg inn i dokumentasjonen av eksisterende system før dette arbeidet starter.

Sammfunspåvirkning

Som ingeniør er det viktig å ta stilling til profesjonsetiske spørsmål knyttet til arbeidet man gjør. Vil produktet ha negative konsekvenser på samfunnet? Er arbeidstakeren en organisasjon man kan støtte med god samvittighet? Kan våre funn legge grunnlaget for utviklingen av et farlig produkt? Vår oppgavestiller, Institutt for bygg- og miljøteknikk, anser vi som en pålitelig organisasjon og en positiv bidragsyter til samfunnet. Gjennom arbeidet instituttet gjør bidrar de til teknologiske fremskritt. Produktet er heller ikke utviklet for økonomisk gevinst, men for å dele NTNU-modellen med tunnelindustrien i bytte mot statistikk fra tunnelprosjekt som tar i bruk nettsiden. På denne måten sikrer instituttet videre innsamling av data som vil i fremtiden legge grunnlaget for videre forskning og en mer nøyaktig modell.

Et av målene beskrevet i oppgaveteksten var å skape en løsning for bruk i undervisning av TBM. Med et stort fokus på et intuitivt brukergrensesnitt og språkstøtte for norsk håper gruppen at løsningen treffer dette målet. På denne måten kan løsningen bidra til videre forskning og utdanning innen fagfeltet, og fasilitere for ny teknologi og kunnskap.

Bærekraft

For å vurdere bærekfraften til det produktet fremmer, TBM-prosjekter, vil vi se på en livsløpsvurdering (LCA) av tunnelprosjekter fra 2012: Life Cycle Assessment of Technical Solutions for High-Speed Rail: Tunnel and Track designs"[24]. Denne masteroppgaven sammenligner ulike metoder for tunnelkonstruksjon til bruk av jernbane, inkludert tunnelboremaskiner. Den funksjonelle enheten brukt i denne analysen er 1 meter tunnel, og typen maskin valgt i LCA-analysen er 'Double shielded', en TBM som legger betongelement langs tunnelveggen imens den borer. Rapport sammenligner resultatene ved å se på miljøpåvirkningene 'Bidrag til klimaforandringer', og 'Menneskelig toksisitet'. I forhold til tradisjonell drilling og spregning bidro Double shielded TBM 32% mer til klimaforandringer, og produserte 74% mer menneskelig toksisitet [24, p. 74]. Derimot viser rapporten at ved å ikke legge betongelement vil bidraget til klimaforandringer synke med 64% og menneskelig toksisitet vil synke med 40%, i forhold til hvis den gjorde det [24, Figur 23]. NTNU-modellen for TBM tar utgangspunkt i boring av harde bergarter hvor Main beam TBM er mer relevant, og betong ikke legges siden fjellet er rigid nok. Et estimat for Main beam TBM blir dermed rundt 52.5% mindre bidrag til klimaforandringer og 4.4% mer menneskelig toksisitet, i forhold til drilling og spregning i tunnelprosjekter ment for jernbane.

Referanser

[1] F.J. Macias. «Hard Rock Tunnel Boring: Performance Predictions and Cutter Life Assessments». Appendix: The NTNU prediction model for hard rock TBMs. Doctoral thesis. 2016. ISBN: 978-82-326-2043-2.

[2] J.V. Thue. *Tunnelboremaskin*. URL: https://snl.no/tunnelboremaskin. (Hentet: 28.03.2023).

[3] The Robbins Company. *Main Beam*. 2021. URL: https://www.robbinstbm.com/products/tunnel-boring-machines/main-beam/. (Hentet: 15.05.2023).

[4] Washington State Dept of Transportation. *The almost finished product*. Lisens: (CC BY-NC-ND 2.0). URL: https://www.flickr.com/photos/wsdot/8260834957/in/album-72157675202550505/. (Hentet: 28.03.2023).

[5] A. Bruland. *Project Report 1B-98: HARD ROCK TUNNEL BORING Advance Rate and Cutter Wear*. Appendix D: Estimation Forms. 1998.

[6] B.R. Bakshi. *Sustainable Engineering Principles and Practice*. Cambridge University Press, 2019.

[7] Cheapwarez. *FullProf: a program for full profile drilling*. URL: https://cheapwarez.com/anleggsdata/anleggsdata-fullprof-a-program-for-full-profile-crack-serial-software-download/. (Hentet: 14.04.2023).

[8] A. Røen og H. Farstad. *Et system for kalkulering av prognosemodell for inndrift ved tunnelboring*. URL: http://www.iie.ntnu.no/prosjekter/bacheloroppgave/v2017/7e/007E.pdf. (Hentet: 14.04.2023).

[9] S. Grønmo. *Kvantitativ metode*. URL: https://snl.no/kvantitativ_metode. (Hentet: 18.05.2023).

[10] S. Grønmo. *Kvalitativ metode*. URL: https://snl.no/kvalitativ_metode. (Hentet: 18.05.2023).

[11] B.J. Oates. *Researching Information Systems and Computing*. Sage Publications, 2006.

[12] P. Borrelli. *Angular vs. React vs. Vue.js: Comparing performance*. URL: https://blog.logrocket.com/angular-vs-react-vs-vue-js-comparing-performance/. (Hentet: 21.05.2023).

[13] Stackshare. *MySQL*. URL: https://stackshare.io/mysql. (Hentet: 14.04.2023).

[14] MVN Repository. *Maven Repository: Top Projects*. URL: https://mvnrepository.com/popular. (Hentet: 20.05.2023).

[15] auth0. *Libraries for Token Signing/Verification*. URL: https://jwt.io/libraries. (Hentet: 27.03.2023).

[16] H.I. Frostad et al. *NTNU TBM model 2016, Excel based estimation tool for prediction of hard rock TBM performance, unpublished*. 2021.

[17] WebAim. *Contrast Checker*. URL: https://webaim.org/resources/contrastchecker/. (Hentet: 12.05.2023).

[18] OWASP. *OWASP Top 10:2021*. URL: https://owasp.org/Top10/. (Hentet: 18.05.2023).

[19] Auth0. *Token Best Practices*. URL: https://auth0.com/docs/secure/tokens/token-best-practices. (Hentet: 27.03.2023).

[20] Spring. *Password Storage*. URL: https://docs.spring.io/spring-security/reference/features/authentication/password-storage.html. (Hentet: 27.03.2023).

[21] i18next. *Introduction - i18next Documentation*. URL: https://www.i18next.com/. (Hentet: 19.05.2023).

[22] OWASP. *Password Storage*. URL: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html. (Hentet: 27.03.2023).

[23] Statcounter. *Browser marketshare worldwide*. URL: https://gs.statcounter.com/browser-market-share. (Hentet: 15.05.2023).

[24] A.M Lia. *Life Cycle Assessment of Technical Solutions for High-Speed Rail: Tunnel and Track designs*. 2012.

Vedlegg

Vedlegg A til F er vedlagt i Inspera:

Vedlegg A: Forprosjektplan

Vedlegg B: Prosjekthåndbok

B1: Møte-innkallelser og -referat

B2: Timeliste

B3: Arbeidskontrakt

Vedlegg C: Visjonsdokument

Vedlegg D: Kravdokumentasjon

Vedlegg E: Systemdokumentasjon

Vedlegg F: WCAG-sjekkliste

Vedlegg G: Sprint Planning

Sprint 2

Funksjon	Prioritet
Brukersystem	1
Lage og dele prosjekter	1
Ferdigstille og kvalitetssikre modellen	1
Soner	1
Modellering av usikkerhet	2
Administratorpanel	2
Forbedre brukergrensesnitt i dashboard	2
Graf av beregningsresultat	3
Andre modeller	3
Forbedre parameterinfo	3
Nye input typer	3
Oppdatere landingsside	4

Sprint 3

Funksjon	Prioritet
Ferdigstille og kvalitetssikre modellen	1
Soner	1
Anbefalt mot faste verdier	1
Modellering av usikkerhet	2
Administratorpanel	2
Nye input typer	2
Forbedre brukergrensesnitt i Dashboard	3
Forbedre parameterinfo	3
Graf av beregningsresultat	4
Andre modeller	4
Oppdatere landingsside	4
Eksportere til PDF	4

Sprint 4

Funksjon	Prioritet
Publisere siden offentlig	1
Epost bekreftelse/invitasjoner	1
Loading/optimalisering	1
Anbefalt og faktisk verdi	1
Brukervilkår	1
Admin lage ny admin	1
Usikkerhet	3

Vedlegg H: Sprint Review

Sprint 1 Review

Dato: 21.03.2023

- For mange parametere vises på én gang, vis det essensielle.
- Machine data og Geologi er viktige kategorier, kanskje vis det først.
- Det var ikke åpenbart at det var kategori-faner.
- Min og max verdi er bare relevant på input-siden til venstre. På høyre i resultat er ikke min og max relevant siden dette er et resultat.
- Correction factors er ikke interessant innledningsvis.
- Vis mer tydelig at resultat-boksen er resultat, og at tallverdiene ikke kan endres.
- Pilene i impact tree virker omvendt. Kanskje tegn piler istedenfor streker i grafen.
- Trenger enheter på parameterne i impact tree.
- Trenger definert antall desimal-tall for alle parametere, f.eks. trenger Number of cutters ingen desimaler, siden det er et heltall.
- Relative position of the average cutter skal være 0.59. Funksjonen som gir den som output må fjernes. Parameteren skal bare være input.
- Det er 5 sentrale parametere i modellen, de burde fremheves og mulig vises alltid.
  - Mm per omdreining
  - Meter per time
  - Machine utilization
  - Meter per uke
  - m^3 per cutter
- Alle konsekvensene av en parameter-redigering er ikke veldig viktig, kan gjemmes vekk på et vis og vises bare hvis man vil.
- Det bør være en hard-limit for parametere, nekt utregning istedenfor å bare vise advarsel.
- For graf-komponenten hadde det vært interessant å kunne velge input parametere som indirekte har en effekt på en gitt output, ikke bare direkte input. For eksempel hvordan TBM Diameter har en effekt på Basic Net Penetration Rate.
- "My Projects" ikke "Your Projects" i prosjekt-oversikt.
- Beregning av kostnader skal kuttes ut. Fjern fra visjonsdokument.
- Legg ved link til doktoravhandling pdf i nettsiden.
- Tunnel length er 1 til 22 km, ikke 0 til 25 km.

Sprint 2 Review

Dato: 11.04.2023

- 'Mine' ikke 'my' i prosjekt-oversikt
- Ønsker å sortere prosjekt-oversikt på forskjellige kolonner
- Ønsker å se 'Last modified' istedenfor 'Created' i prosjekt-oversikt
- Vis prosjekt-tittel inne i prosjektet, kanskje vise 'Created' også inne i prosjektet
- 'Project title' istedenfor 'Name' i prosjekt-oversikt
- Skriv 'Add project' i tillegg til + icon for å lage nytt prosjekt, dette er mer tydelig
- Trenger navn i bruker-objektet, vil heller se navnet til andre enn e-post i frontend
- Ønsker å se hvem som er i ett prosjekt du er invitert til
- Strøk og fall kalkulator for å finne vinkel i Fracture Set
- Egen + og - knapp for fracture set for å legge til/slette
- Ønsker å sette navn på soner. Vis både zone-index og navn.
- Sum av alle sone-lengder må være lik tunnel-lengde, lag en sjekk på dette.
- Solid cubic meters, basic net penetration rate, og weekly advance rate vises som resultat i soner-vinduet. Resten kan vises i avansert-modus.
- Lag kopi av TBMMath.recommendedGrossAverageThrustPerCutterDisc() for å bruke i soner, der input er fracture factor istedenfor fracture class. Interpoler fra 0.36 til 4.
- Max recommended cutter thrust for maskin istedenfor Gross average cutter thrust. Gross average cutter thrust for soner kan maks være max fra maskin. Standard: 300 kn/cutter
- Det samme gjelder RPM, sett en maks-verdi i prosjekt som begrenser RPM i soner.
- By default skal recommended verdi brukes for alle parametere. Når bruker overstyrer, skal actual verdi brukes. Når en parameter endres som følge av en annen parameter er det bare recommended verdi som endres.
- Når en overskrevet verdi blir den samme som en anbefalt verdi, skal den overskrevede verdien fjernes og parameteren går tilbake til å bruke anbefalt verdi. Som om brukeren aldri hadde overskrevet.
- Ønsker at MSJ parametere vises som input etter m/h i soner, deretter kommer videre resultat under der igjen.
- Ønsker at totale resultat fra soner vises som en kolonne til høyre.
- Øsnker at når man trykker rett på en kategori istedenfor til venstre/til høyre, skal bare denne kategorien vises.

Sprint 3 Review

Dato: 24.04.2023

- Mangler fortsatt prosjekt-tittel i Navbar
- Rund av til korrekt antall desimaler i soner-vindu
- Ønsker at fracture class og orientation bytter plass i sone-vindu
- Ønsker fortsatt total-kolonne i soner-vindu
- Noe går feil når man dupliserer en sone, finn bug.
- Parameters i admin-panel funker ikke
- DRI og CLI er ikke prosent, ikke oppgi enhet på disse.
- Ønsker tykk strek på midtverdien for DRI/CLI i steintype-vinduet.

Vedlegg I: Backlog

Issue ID	Title	Milestone
1	Projects overview page	Sprint 1
2	User page	Sprint 1
3	Landing page	Sprint 1
4	User help page and Learning page (Link in footer)	Sprint 1
5	Impact tree	Sprint 1
6	Graph card	Sprint 1
7	Notification card on dashboard	Sprint 1
8	Login page	Sprint 1
9	Register page	Sprint 1
10	Norwegian language support	Sprint 3
11	Administrate parameters page	Sprint 1
12	CI Backend	Sprint 1
13	CI Frontend	Sprint 1
14	Implement backend model	Sprint 1
15	Dashboard page	Sprint 1
18	Implement framework for creating and updating the actual TBM model.	Sprint 1
19	Implement category in the backend and frontend	Sprint 1
20	Framework for language support	Sprint 1
21	Use locale in frontend	Sprint 1
22	Dark/Light mode	Sprint 1
23	Move dark/light mode from App.vue to seperate css files	Sprint 1
24	Setup flyway	Sprint 1
25	Port frontend to typescript	Sprint 1
26	Implemented API-documentation	Sprint 1
27	Implementing TBM-math functions	Sprint 1
28	Wire functions in sql (Description for list)	Sprint 1
29	Endpoint for updating locale values	Sprint 1
30	Add locale keys for parameter names	Sprint 1
31	Implement parameter priority	Sprint 1
32	Fix issue where impact list contains duplicates	Sprint 1
33	Fix search in frontend	Sprint 1
34	Implement Unit in the backend	Sprint 1
35	Implement Notation in the backend	Sprint 1
36	Error handling on parameter input	Sprint 1
38	Add locale keys for error messages	Sprint 1
39	Zone card prototype	Sprint 2
40	Add initial max and min values	Sprint 1
41	Add template project	Sprint 1
42	Change language frontend	Sprint 1
44	Revisjonshistorie forprosjektplan	
45	Oppstart hovedrapport	Sprint 1
46	Fix function for basic penetration rate + various frontend fixes	Sprint 1
47	Remove warning when parameter is back in range	Sprint 1
49	Parameter information card	Sprint 4
50	Move parameter input to the right	
51	Remove waste of space between navbar and dashboard	
52	Increase max height of Edit/Result cards, compress data	
69	Search results stays when going out and in of project.	Sprint 2
70	Fix value limits for Tunnel Length	Sprint 2

71	Sprint 1 feedback fixes/additions	Sprint 2
72	Add limits to parameters	Sprint 2
75	Fjern kostnader fra visjonsdokument	Sprint 2
76	Screenshots of sprint 1 frontend	
77	Authentication and authourization	Sprint 2
78	Connect user profile	Sprint 2
79	Create, update, duplicate and delete project	Sprint 3
80	Share projects with other users	Sprint 2
81	Quality assurance of the implemented model.	Sprint 4
82	Implement geological zones	Sprint 3
83	Modellation of uncertainty in paramter values.	Sprint 4
84	Monte Carlo simulation	Sprint 4
85	Administration of parameter names, descriptions and limits.	Sprint 2
86	Administrate text (language)	Sprint 2
87	Dashboard redesign	Sprint 4
88	Redesign parameter info	Sprint 4
89	Create new input types	Sprint 3
90	Refine landingpage	Sprint 4
91	Update mathematical model with new values	Sprint 4
92	Function wiring bug	Sprint 3
94	Function wiring bug 2	Sprint 3
95	Circular dependency ProjectParameterService	Sprint 4
96	Recommended cutterhead RPM is set to 0 initally	Sprint 3
97	Publish app to production	Sprint 4
98	Total results column in Zone view	Sprint 4
99	Add input fields for MSJ parameters in Zones view	Sprint 3
100	Strike and dip angle calculator	Sprint 4
101	Help page	Sprint 4
102	Add loading and button disable	Sprint 4
103	Recommended vs Actual values	Sprint 4
104	Terms of service	Sprint 4
105	Admin panel roles	Sprint 4
106	Frontend loading on requests	Sprint 4
107	Fix parameters in admin panel	Sprint 4
108	Access token is not seamlessly refreshed when it expires naturlally.	Sprint 4
109	Category locale	Sprint 4
110	Merge 'zones'-tab into 'project'-tab	Sprint 4
111	Implement email system	Sprint 4
112	Fix rerouting when logged in and not logged in	Sprint 4
113	Admin panel available in mobile view	Sprint 4
114	Restrictions for Angle Calculator	Sprint 4
115	Project Category	Sprint 4
116	Total results not editable	Sprint 4
118	Edit zone and fracture set params in admin panel	Sprint 4
119	Code coverage	Sprint 4
120	Unit-testing	
122	Dashboard categories at mobile size	
123	Help page mobile/desktop reponsiveness fix	
124	Terms of service should open new tab in register	
125	WCAG 2.1 Fixes	

Vedlegg J: Oppgavetekst

Arbeidstittel:	NTNU TBM - videreutvikling

Hensikten med oppgaven:
Videreutvikle et nettbasert verktøy for prediksjon av produksjon og kostnad ved fullprofilboring av tunneler, se for eksempel therobbinscompany.com.

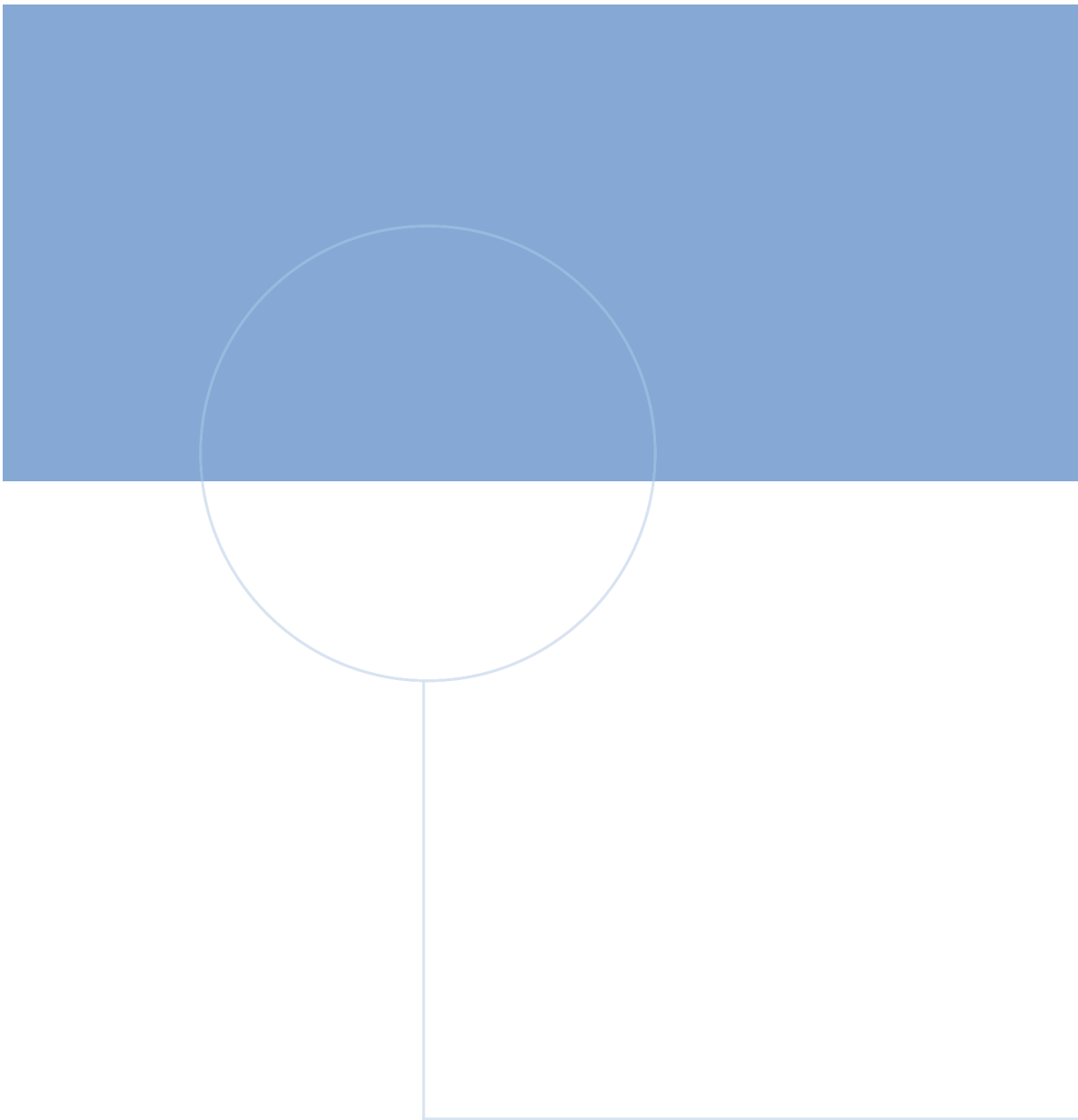
Kort beskrivelse av oppgaveforslag:
NTNU TBM ble utviklet gjennom en BSc-oppgave i 2017.
http://www.iie.ntnu.no/prosjekter/bacheloroppgave/v2017/7e/007E.pdf
NTNUs TBM-modell er mye brukt i internasjonal tunnelbransje. Vi ønsker oss et brukervennlig og nettbasert verktøy som baseres på Javier Macias sin phd-avhandling. Verktøyet som ble utviklet i 2017 har noe av den ønskede funksjonaliteten, men det gjenstår mye på analyse, risiko med mer. Også integrering av andre beregningsmodeller enn NTNU sin.
At verktøyet legger til rette for nettbasert samhandling, er særlig viktig i tidligfasen av tunnelprosjekter, siden TBMteknologien
og -bransjen er internasjonal.
Det er ønskelig at applikasjonen kan simulere usikkerhet i beregningsmetode (ulike beregningsmodeller) og i inngangsparametere.

Utfyllende kommentarer til hva oppgaven gjelder:
NTNU TBM i sin første versjon, er en web- og databasebasert løsning.
Det vil bli gitt en grundig innføring i fullprofilboring av tunneler som metode, samt god og tett oppfølging og veiledning undervegs.
Demovideoer av fullprofilboring https://www.youtube.com/watch?v=y7acjVE17zA
https://www.herrenknecht.com/en/products/productdetail/gripper-tbm/

Oppgaven passer for (kryss av de(t) som passer og skriv evt. en kommentar til oss):	- Bacheloroppgave
Hvilket studieprogram og emne passer oppgaven til? (spesifiseres ved bacheloroppgaver)	IDATT2900 – Bacheloroppgave Dataingeniør
Skal oppgaven utføres av bestemte studenter? (der avtalt) Fyll i så fall inn studentenes navn	Nei.
Kan oppgavestiller stille arbeidsplass med nødvendig utstyr og programvare:	Nei.
Hvis ikke, hva kreves av maskin og programvare:	Ikke noe spesielt.
Begrensninger i tilgjenglighet av opplysninger o.l:	Ingen begrensninger. NTNU-modellen for TBM-boring er publisert internasjonalt.
Oppgaven passer best for, antall studenter:	- 2 - 3

Opplysninger om oppgavestiller

Er du fra en bedrift/virksomhet eller er du student med en egendefinert/selvlaget oppgave?	- Bedrift/virksomhet



NTNU

Kunnskap for en bedre verden