Høgskulen på Vestlandet

BACHELOROPPGAVE

Tilgjengeliggjøre teknisk data og dokumentasjon fra CIS plattformen for brukere i felt

Make technical data and documentation available from the CIS platform for users in the field

Sander Andre Berg Marx, Chris Even Furdal Lyngholm, Ali Abdi Nur Ali

Bachelor, Dataingeniør
Institutt for data- og realfag
Fakultet for ingeniør- og naturvitenskap
Veileder: Rogardt Heldal

4. Juni, 2021

TITTELSIDE FOR HOVEDPROSJEKT

Rapportens tittel: Tilgjengeliggjøre teknisk data og dokumentasjon fra CIS plattformen for brukere i felt	*Dato: 04.06.2021*
Forfatter(e): Sander Andre Berg Marx, Chris Even Furdal Lyngholm, Ali Abdi Nur Ali	*Antall sider u/vedlegg: 40* *Antall sider vedlegg: 12*
Studieretning: Dataingeniør	*Antall disketter/CD-er: 0*
Kontaktperson ved studieretning: Rogardt Heldal	*Gradering:*Ingen
Merknader:	

Oppdragsgiver: Sharecat Solution AS	*Oppdragsgivers referanse:*
Oppdragsgivers kontaktperson: Alexander Algaard	*Telefon:* 90604899

Sammendrag: Prosjektet handlet om å tilgjengeliggjøre teknisk data og dokumentasjon fra Sharecat sin CIS plattform til brukere i felt, dette ble gjort med å lage en progressiv web applikasjon (PWA) som kan kjøre som en applikasjon på mobile enheter. Prosjektet er utviklet med tanke på mobile enheter og har et mobilt brukervennlig brukergrensesnitt.

The project consisted of making technical data and documentation from Sharecats CIS platform available for users in the field, this was done by developing a progressive web app (PWA) that can run as a mobile application. The project is developed with mobile devices in mind and has a mobile friendly user interface.

Stikkord:

Elasticsearch	React	Progressive Web App (PWA)

Høgskulen på Vestlandet, Fakultet for ingeniør- og naturvitskap
Postadresse: Postboks 7030, 5020 BERGEN Besøksadresse: Inndalsveien 28, Bergen
Tlf. 55 58 75 00 Fax 55 58 77 90 E-post: post@hvl.no Hjemmeside: http://www.hvl.no

Forord

Når vi skulle velge bacheloroppgave, var det oppgaven fra Sharecat som stakk seg litt ut. Her fikk en mulighet til å utforske mange interessante områder som mobil applikasjonsutvikling, håndtere store mengder data og lage et funksjonelt brukergrensesnitt med HTML, CSS og JS. Det kom allikevel frem underveis i oppgaven at den mulig var litt mer ambisiøs enn hva vi hadde muligheter til å levere. Til tross for dette, klarte gruppen å levere en prototype som hadde de fleste funksjoner som oppdragsgiver ønsket

Oppgaven har selvfølgelig vært mer krevende dette semesteret hvor både gruppen og våre veiledere har måtte gjøre et bytte fra fysisk oppmøte til kun digitalt. Dette merket vi nok mer i starten på oppgaven hvor alt tok litt lengre tid, men etter et par uker begynte det å komme seg og det ble mye bedre flyt i arbeidet.

Etter dette ble det en sakte men sikker fremgang hver uke hvor applikasjonen og ikke minst den akademiske oppgaven begynte å ta mer form. Hele prosessen har vært en svært lærerik opplevelse og vi har fått muligheten til å øke vår kompetanse som utviklere og fått et bedre innblikk i hvordan programutvikling fungerer i en jobbsammenheng.

Vi ønsker å takke våre oppdragsgivere Lars Sørås og Alexander Algaard samt vår veileder Rogardt Heldal. De har stilt opp hver uke noe som har bidratt både til å øke vår forståelse for prosjektet og gjort at arbeidet med den akademiske oppgaven og utviklingen har vært mye enklere.

1 Innledning

For å kunne utføre nødvendige installasjoner og utbytting av utstyr på oljeplattformer, har Sharecat AS et behov for å gi operatører tilgang på dokumentasjon. De har i dag en løsning som fungerer som en desktop applikasjon, men har ingen løsning for mobile enheter. Ettersom operatører ofte ikke har tilgang til desktop versjonen når de er i arbeid, er det et stort behov for å få utviklet en applikasjon som fungerer på mobile enheter.

Dette arbeidet involverer å utvikle en helt ny applikasjon, men som tilgjengeliggjør samme data som deres nåværende løsning. Dette krever arbeid innen databasehåndtering, utvikle underliggende logikk som sikkerhet og videresending av data samt lage et passende brukergrensesnitt. Denne applikasjonen består i hovedsak av 3 komponenter som skal ha flyt av informasjon med bruk av en lagvis tilnærming og kalt «Layered Approach».

1.1 Mål

Målet med prosjektet er å utvikle en mobil applikasjon som gir tilgang til Sharecat sin CIS Platform for brukere i felt. Dokumentasjonen fra CIS plattformen vil bli tilgjengelig gjennom en søkemotor hvor en kan søke på teknisk informasjon basert på utstyrsidentitet og/eller relevante nøkkelord.

Applikasjonen må være enkel å bruke og ha et spisset brukergrensesnitt for å tilgjengeliggjøre data og dokumentasjon slik at bruker i felt ikke trenger å gå igjennom komplisert opplæring for å bruke applikasjonen. Applikasjonen bør også ha en velfungerende maskinlæringsmodell for bildeprosessering av tekst slik at brukere kan gjøre søk gjennom å ta bilde av en streng med bokstaver.

1.2 Kontekst

Problemet som denne rapporten skal utforske er. Hvor bra kan gruppen tilgjengeliggjøre teknisk data og dokumentasjon fra CIS plattformen til Sharecat for brukere på mobile løsninger i form av en mobil app.

For å utvikle en bra mobil løsning så bør applikasjonen være lett å bruke. Ha et enkelt system for å logge inn og ut, der en bruker kan enten være personell eller inspektør. Den skal ha et effektivt og enkelt søkegrensesnitt som gjør at brukere kan lett finne frem relevant dokumentasjon. Brukere skal kun kunne lese av data i appen, de skal ikke ha mulighet å skrive noe data inn i dokumentasjonen. Dokumentasjonen hentes fra en server og vises i pdf format.

1.3 Ressurser

For å kunne utvikle applikasjonen i henhold til oppgaven, trenger gruppen tilgang på en rekke materielle og immaterielle ressurser som data, kunnskap, software, hardware og veiledere. For data, er det i denne applikasjonen i hovedsak en dokumentdatabase for uthenting av filer, brukerinformasjon for å validere og autorisere tilgang til applikasjonen og en database av tagger (identifikatorer) for å kunne finne frem til rett dokument.

Gruppen har også et behov for veiledning og kunnskap gjennom prosjektet, som her i hovedsak er begrenset til prosjektets oppdragsgivere Alexander Algaard og Lars Sørås. De har stått for å forsikre at prosjektet får den kvaliteten og funksjonaliteten som er ønsket. De har også bidratt med ressurser i form av virtuelle maskiner med databasetilgang som gruppen har fått tilgang til å bruke.

Når det gjelder software og hardware, vil en i kapittel 3 komme tilbake til dette.

2 Prosjektbeskrivelse

I denne delen, vil en gå over spesifikasjoner rundt utviklingen av en IOS basert applikasjon basert på ønsket funksjonalitet og krav fra oppdragsgiveren Sharecat. Denne applikasjonen vil fungere som et verktøy for ingeniører på oljeplattformer som har et behov for umiddelbar tilgang på dokumentasjon.

2.1 Praktisk bakgrunn

2.1.1 prosjekteier

Sharecat er ett programvare- og konsulentselskap som tilbyr tjenester for håndtering av teknisk leverandørinformasjon. Sharecat spesialiserer seg i å levere løsninger til industrier som olje og gass, petrokjemi og kraft. Selskapet ble etablert i 1993 og har omtrent 50 ansatte med kontorer i Bergen og tilstedeværelse i Storbritannia og Malaysia (Linkedin, 2021).

Sharecat sin CIS plattform er et verktøy for håndtering av data i komplekse petrokjemiske prosjekter. Løsningen har blitt brukt på mange av de største utbyggingsprosjektene både på norsk sokkel og internasjonalt.

Mer enn 30 store olje og gass aktører har brukt Sharecat til forbedringsprogrammer og katalogeringsprogrammer for anlegg som dekker mer enn 100 eiendeler. Sharecat har lenge jobbet tett med de største olje og gass aktørene noe som har gjort at Sharecat sine løsninger har utviklet seg til å bli pålitelige og robuste.

2.1.2 Tidligere arbeid

Ettersom Sharecat allerede hadde en eksisterende desktop applikasjon, var det mulig å studere denne, noe som har bidratt til å inspisere designet på den mobile versjonen. Det var likevel et behov for å utvikle applikasjonen fra bunnen av slik at det ikke var mulighet for gjenbruk av kode. Det som kunne gjenbrukes, var deres eksisterende databasesystem. Dette gjorde at mye av databasearbeidet var begrenset til uthenting av data.

2.1.3 Initielle krav

Kravene til applikasjonen, bygget i hovedsak på deres desktop versjon, men med utvidet funksjonalitet og et mer brukervennlig brukergrensesnitt. Dette innebærte at i tillegg til uthenting av filer og visning av dokument attributter, hadde de også ønske om å kunne lese av fysiske tagger (identifikatorer) med bruk av bildeprosessering og maskinlæring. For brukergrensesnittet, trengte en et nytt og intuitivt design som fungerer godt med touch.

De resterende kravene, er listet nedenfor.

- Innloggingssystem med bruk av Office 365 føderering
- Utflating av Sharecat's ElasticSearch database
- Lagring av dokumenter internt på mobile enheter
- Uthenting av dokumenter og visning av pdf filer
- Utpakking av zip filer
- Filtrering av hvilke dokument attributter som skal vises
- Søkemotor basert på tag (identifikator)
- Database med emailliste for føderert innlogging

2.1.4 Initiell løsnings-idé

Hovedtanken bak løsningen, var å ta deres eksisterende desktop applikasjon og redesigne den med tanke på brukervennlighet på mobile enheter. Dette krever at en har en 3-delt applikasjon, hvor en har en database med informasjon, en underliggende logikk (backend) og en frontend for brukergrensesnitt og visning av data.

For selve utformingen av applikasjonen, var ideen å ha en søkbar hvor brukerne kan skrive inn deler eller hele navnet på dokumentet en ønsker og så vil applikasjonen i gjengjeld vise resultatene til brukeren. Etter at brukeren har sett hvilke filer som matcher søket, vil det være en mulighet å se nærmere på dokumentet for å se attributter, eller å vise aktuell fil i en pdf viser.

Det skal også være mulig å søke med bruk av en tag (identifikator) enten hvor brukeren skriver inn taggen manuelt, eller at en bruker kamera på den mobile enheten til å skanne en tag og i gjengjeld vil applikasjonen kjøre et søk basert på taggen og gi matchende resultatet.

3 Design av prosjektet

I denne delen, vil en gå over ulike løsningsalternativer for applikasjonen, hvilke løsning og verktøy som ble valgt, samt ulike metoder for utvikling og prosjektet.

3.1 Forslag til løsning

For dette prosjektet, var det begrenset hvilke løsninger en kunne bruke ettersom oppdragsgiver hadde konkrete ønsker rundt applikasjonens design og rammeverk som skulle brukes. Det ble allikevel gjort en vurdering om applikasjonen skulle bli laget som en native IOS applikasjon, eller en PWA (progressive web app) bygget på HTML, CSS og JS.

3.1.1 Alternativ 1 - Progressive web app

Denne løsningen består av å lage en webbasert applikasjon basert på HTML, CSS og JS. I tillegg så er det et behov for underliggende logikk kodet i Java som kan ta seg av sikkerhet, database kommunikasjon og andre integrasjoner.

Denne typen applikasjon vil simulere en webbasert applikasjon noe som gjør at den kan lanseres på mange plattformer og operativsystemer med en kodebase. Det er allikevel noe begrensinger når det kommer til å utnytte alle funksjoner av hvert operativsystem.

3.1.2 Alternativ 2 - Native IOS applikasjon

Ettersom sluttbrukerne i hovedsak bruker IOS enheter, ville en mulighet være å utvikle en mobil applikasjon i Swift. Her ville både underliggende logikk samt brukergrensesnittet være designet i samme språk. En slik applikasjon kan bruke alt av tilgjengelige funksjoner av operativsystemet, men selve utviklingen kan være noe vanskeligere. Denne løsningen går og på bekostning av at en ikke kan lansere applikasjonen på andre plattformer enn IOS om det skulle bli relevant.

3.2 Valgt Løsning

Den valgte løsningen endte opp med å bli en progressive web app ettersom denne løsningen hadde mange komponenter og rammeverk som gruppen allerede var noe kjent med. Kodespråket til den underliggende logikken (Java) var også kjent i motsetning til om en skulle hatt en native app.

En PWA blir også ansett som lette å utvikle og gir større fleksibilitet enn en native app, selv om dette kan gå på bekostning av noe tapt native funksjonalitet. Denne løsningen var også også et ønske fra oppdragsgiver noe som videre forsterket valget.

3.3 Valg av verktøy

3.3.1 Software

Visual studio code:

Dette er en IDE (integrated development environment), som er et utviklingsmiljø for programvare hvor en har samlet mange ulike utviklingsverktøy i et brukergrensesnitt. Her var det flere alternativer å velge mellom, men hvilken en velger, har mest med utviklernes preferanser. Etter å ha vurdert flere utviklingsmiljøer, ble gruppen enige om at for dette prosjektet, passet visual studio code best.

Elasticsearch:

Ettersom dette prosjektet angår uthenting og visning av filer, har oppdragsgiver gitt gruppen tilgang en indeks (samling av data) i en Elasticsearch database. Dette er en type database som i hovedsak blir brukt når en trenger en dokumentbasert lagring (pdf filer) som har et behov for en mer dynamisk lagring med mindre strenge krav. Her var det ikke mulighet for å velge noe annet, så dette måtte gruppen ta i bruk.

Docker:

For å distribuere applikasjonen, hadde oppdragsgiver ønske om å enkapsulere applikasjonen i en container slik at den enkelt kan kjøres på mange ulike enheter og operativsystemer. Oppdragsgiver nevnte her bruk av Docker som et verktøy til å utføre denne jobben, noe gruppen tok utgangspunkt i. Dette verktøyet fungerer med at en lager en kjørbar docker fil, som inneholder all applikasjonsdata og nødvendige avhengigheter.

React:

Når det gjelder å utvikle et brukergrensesnitt for applikasjonen, valgte gruppen å bruke et JavaScript bibliotek kaldt React. Dette er en komponentbasert tilnærming til frontend utvikling basert på HTML, CSS og JavaScript. Dette ble i hovedsak valgt ettersom det er enkelt å lære og bruke, samt har blitt er en av industristandardene for frontend utvikling. Her var det andre alternativer også, men enkelte av gruppemedlemmene hadde allerede noe kunnskap om dette biblioteket, så det ble et naturlig valg.

Spring Boot:

For å håndtere den underliggende logikken for applikasjonen, valgte gruppen å ta i bruk Spring Boot. Dette er et javabasert rammeverk som tar seg av mye av forberedelsene og konfigureringen som en ellers måtte ha gjort manuelt når en starter et nytt prosjekt. For å komme raskt i gang med prosjektet, ble derfor dette rammeverket valgt.

SQLite:

Ettersom det i denne applikasjonen er et behov for å kunne vise frem og hente frem filer når applikasjonen ikke har tilgang til internett, var det et behov for en form for midlertidig intern lagring. SQLite ble derfor valgt for å håndtere dette problemet ettersom det gjør det mulig å ha en intern database som ikke krever egen server og tilkobling.

3.3.2 Hardware

Mobile enheter:

For kjøring av applikasjonen, hadde gruppen behov for tilgang til ulike mobile enheter for manuell testing eller gjennom en leverandør som kan utføre mer omfattende kompatibilitetstester.

Stasjonær/Bærbar datamaskin

For selve utviklingen av applikasjonen, ble det i hovedsak brukt stasjonære/bærbare datamaskiner. Dette var noe gruppen fra prosjektstart allerede hadde tilgang til.

Server:

Ettersom applikasjonen krever uthenting av data over nett gjennom en database, kreves det en server hvor denne databasen kan kjøre. Dette er noe oppdragsgiver har gitt gruppen tilgang til som de har satt opp på deres side grunnet datasikkerhet og for å ikke overkomplisere prosjektet.

3.4 Prosjektmetodikk

3.4.1 Utviklingsmetodikk

For utviklingen av prosjektet, ønsket oppdragsgiver at gruppen ikke skulle bruke Scrum, det ble derfor en enighet om å bruke Scrumban. Dette var ettersom Scrum ikke var agil nok ifølge oppdragsgiveren. Scrum er i hovedsak en iterasjon basert utviklingsmetodikk, mens kanban fokuserer på kontinuerlig flyt av arbeid.

Det som er innarbeidet fra Scrum, er at arbeidet er iterasjonsbasert hvor en har en mengde gjøremål i hver sprint (her 1 uke) basert på kompleksitet. En kan også endre prioriteringer underveis basert på hvilken oppgave som blir ansett som viktigst. Det brukes også lister for å holde styr på alle gjøremål og aktive gjøremål.

Fra Kanban, har en mulighet til å hele tiden hente inn gjøremål om det er kapasitet. En har også en begrensning på hvor mange gjøremål som kan være aktive på en gang. Det er også ikke klare roller spesifisert for gjøremålene slik at en kan jobbe hvor det trengs.

Denne kombinasjonen, gir derfor gruppen mulighet til å arbeide iterativt, med god flyt og mulighet til å endre prioriteringer med behov.

3.4.2 Prosjektplan

I dette prosjektet, ble det i første omgang satt opp et møte mellom oppdragsgiver og gruppen slik at det skulle opparbeides en felles forståelse for problemet som skulle løses og hvordan det skulle løses. Etter noe diskutering, ble det enighet om at gruppen skulle utvikle en PWA hvor valgt rammeverk var React for frontend og Spring-boot for backend.

Skissering:

Før selve utviklingen kunne starte, var det viktig med bruk av skisser at felles forståelse mellom gruppen og oppdragsgiver faktisk var nådd. Det skal derfor skisseres et UML og USE-case diagram for applikasjonen.

Veritcal slice:

For å få raske resultater på utviklingen, skal gruppen utvikle en vertikal bit med komponenter fra frontend, backend og databasen. Dette er for å få en kjørbar vertikal bit av applikasjonen som går gjennom alle ledd.

Utvikle nødvendig funksjonalitet:

Her blir arbeidet å utvikle videre funksjonalitet som er nødvendig for at applikasjonen skal fungere.

legge til funksjoner og programvare:

Etter at det nødvendige arbeidet er gjort for at hoved funksjonaliteten skal fungere, blir arbeidet å legge til funksjoner og programvare som øker sikkerhet, ytelse og gjør applikasjonen mer brukervennlig for sluttbrukerne.

Evaluere applikasjonen:

Når applikasjonen har kommet til et punkt hvor det er en testbar prototype med all funksjonalitet som var ønsket, må den testes og evalueres. Det kan deretter bli gjort ønsket endringer etter behov.

Lansering:

Applikasjonen er nå klar til å bli gjort tilgjengelig for mobile enheter og kan bli lastet ned og tatt i bruk.

3.4.3 Risikovurdering

Risikofaktor	S	K	RF	Tiltak	Interessent	Fase
Problemer med å forstå prosjekt oppgaven	5	9	45	Stille presise spørsmål om prosjektet til oppdragsgiver	Gruppen og oppdragsgiver	Utviklingsfasen
Feil Kommunikasjon mellom oppdragsgiver og gruppen	9	3	27	Regelmessige møter hvor en diskuterer fremgangsmåter og prosjektet	Gruppen og oppdragsgiver	Utviklingsfasen
Korrupte filer / mister data	3	7	21	Passe på å ha backup av data	Gruppen, oppdragsgiver og sluttbrukere	Utviklingsfasen
Problemer grunnet digitalt samarbeid	6	3	18	Ha alternative kommunikasjonskanaler	Gruppen og oppdragsgiver	Kontinuerlig
Uenigheter i gruppen skaper forsinkelser	3	2	6	Ha en bestemmelsesprosess om konflikter skulle oppstå	Gruppen og oppdragsgiver	Kontinuerlig
Uferdig / Ikke brukbar applikasjon	3	9	27	God planlegging og holde prosjektplanen	Oppdragsgiver og sluttbrukere	Sluttfasen
Sykdom i gruppen / nøkkelpersoner	5	5	25	Lage en prosjektplan som tar høyde for noen forsinkelser slik at en ikke får for lite tid.	Gruppen og oppdragsgiver	Kontinuerlig
Mangel på brukertesting av applikasjon	7	5	35	Finne brukere som er villig å teste applikasjonen, helst brukere målgruppen	Gruppen og oppdragsgiver	Sluttfasen
Applikasjon ikke kompatibel med nok enheter	6	7	42	Teste applikasjonen på forskjellige enheter	Gruppen, oppdragsgiver og sluttbrukere	Sluttfasen

S: Sannsynlighet K: konsekvens RF: Risikofaktor

Tabell 3.4.1 Risiko liste

3.5 Evalueringsplan

For evaluering og testing av prosjektet, er det viktig at både applikasjonen som en helhet blir testet, men også individuelle komponenter som inngår i applikasjonens interne logikk. Gruppen ville derfor ta i bruk JUnit, Integrasjonstester og mocking for testing og validering av kode biter/deler av systemet. For å teste applikasjonens funksjonalitet og brukergrensesnitt, hadde gruppen et mål om å ta i bruk eksterne testere, enten gitt av oppdragsgiver, eller rekruttert av gruppen.

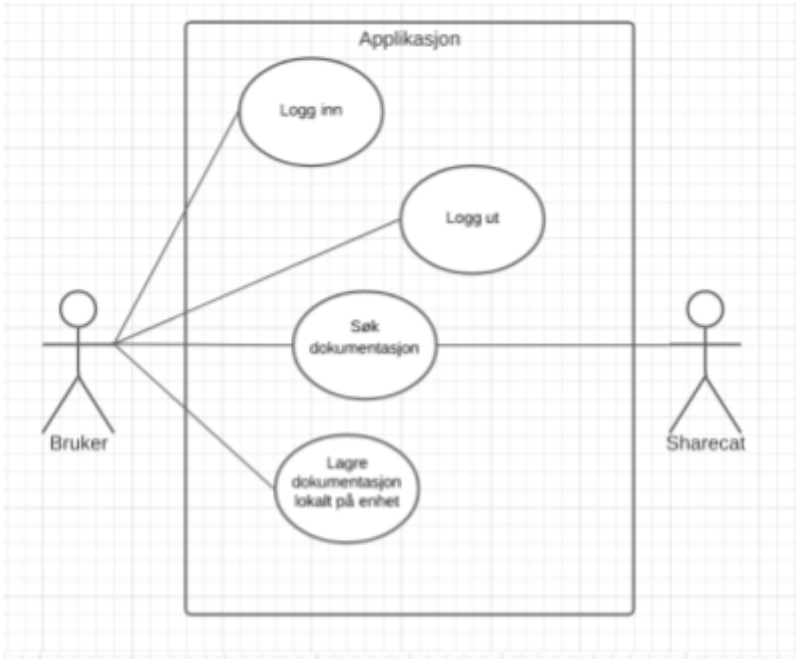
4 Detaljert design

I dette kapittelet blir detaljene i designet bli utforsket. Kapittelet forklarer designet detaljert med å vise UML diagrammer og prosjektets arkitektur. Hvordan appen fungerer og prosjektets forskjellige moduler blir også detaljert.

4.1 UML Diagrammer

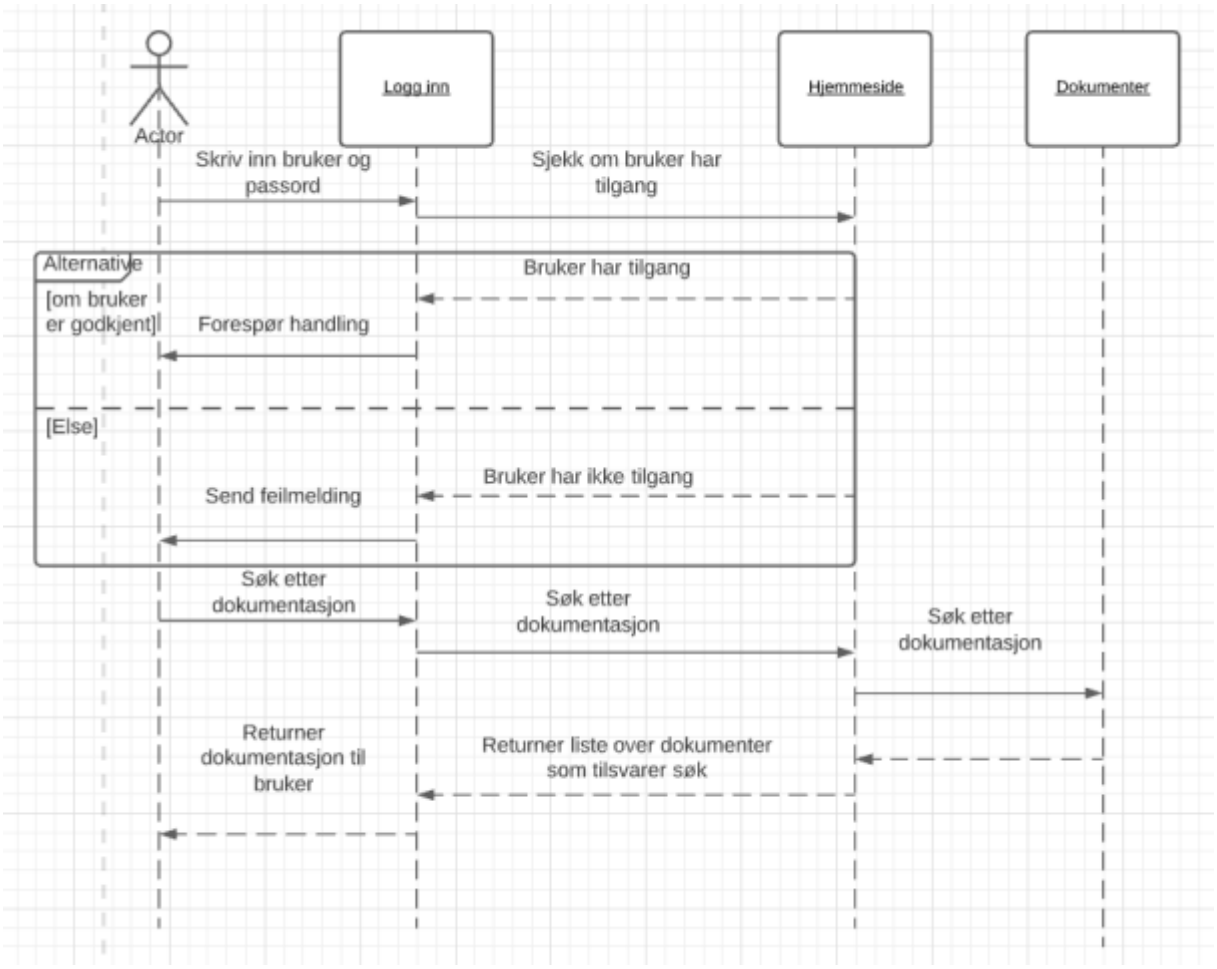
Uml diagrammer gir utviklere en god måte å illustrere hvordan en applikasjon vil virke. Dette gjør det lettere for tredjeparter å forstå designet til applikasjonen og hvordan den vil fungere.

4.1.1 Brukstilfellediagram



Figur 4.1.1 - Brukstilfellediagram som viser de forskjellige handlingene en bruker kan gjøre og viser at Sharecat supplerer dokumentasjonen

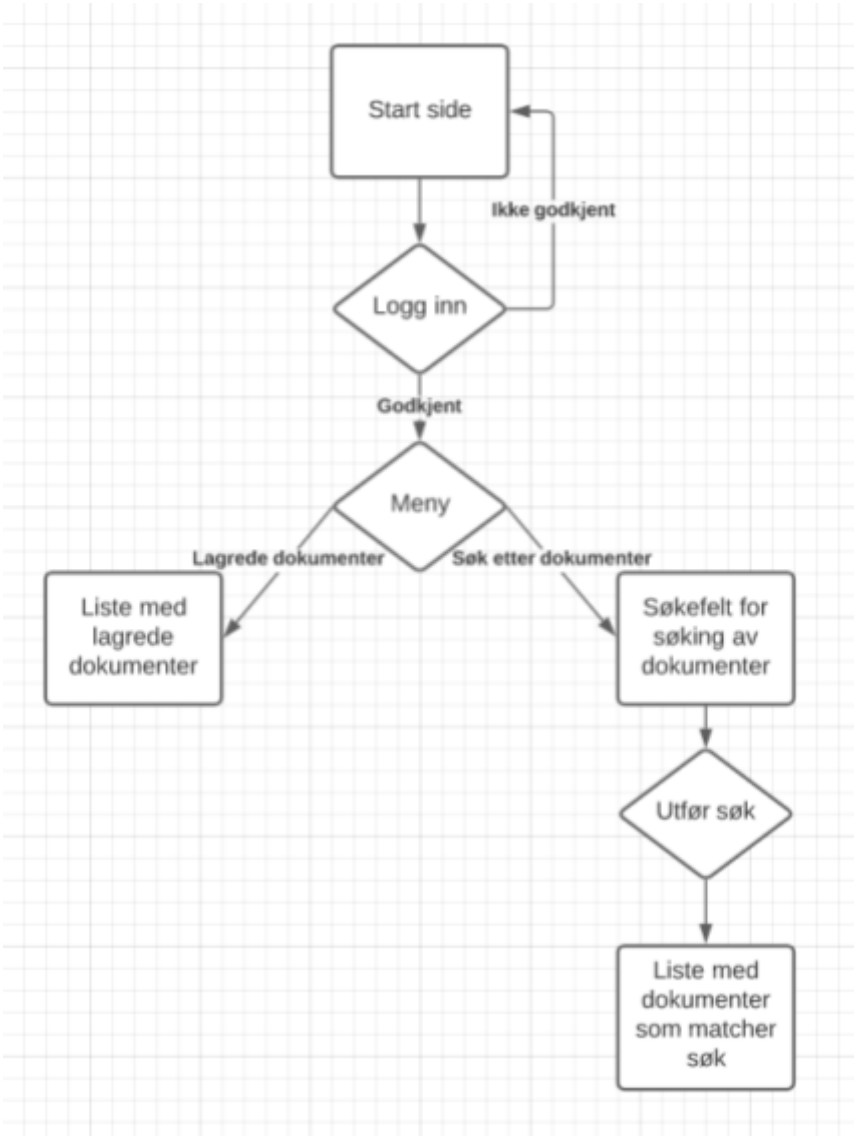
4.1.2 Sekvensdiagram



Figur 4.1.2 - Sekvensdiagram for logg inn av applikasjon og hente dokumentasjon

4.1.3 Flytdiagram

Flytdiagrammet forklarer flyten til applikasjonen, diagrammet illustrerer i prinsipp det samme som sekvensdiagrammet bare på en litt mer oversiktlig og lettleselig måte. Sekvensdiagrammet brukes gjerne for å illustrere hvordan logikken lettere.



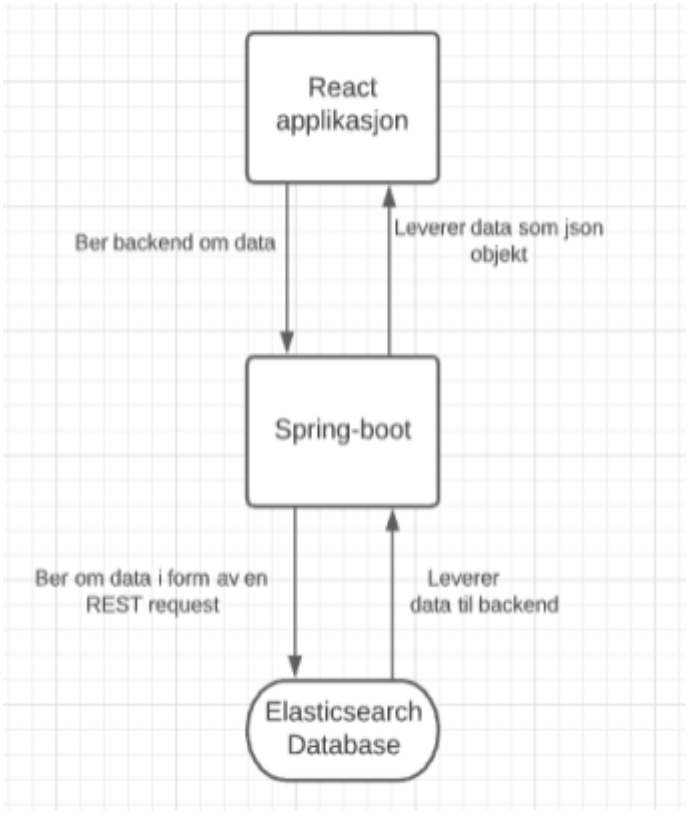
Figur 4.1.3 - Enkelt flytdiagram for applikasjonen

4.2 Arkitektur

Forklaring av arkitekturen til applikasjonen og databasen gjør det lettere å forstå hvordan applikasjonen virker og gjør det lettere å forstå valgene gruppen gjorde når applikasjonen ble designet.

4.2.1 Applikasjonsarkitektur

Applikasjonen ble utviklet som en fullstack-applikasjon, det innebærer en frontend applikasjon som kommuniserer med ett backend API. API henter data fra en database og sender det videre til frontend. Applikasjonen som gruppen har bygget bruker react som frontend, spring-boot som backend og elasticsearch som database. Gruppen valgte å lage en fullstack app på grunn av fleksibiliteten det tilbyr. Ved å dele opp frontend og backend gir det muligheter å gjenbruke backend kode slik at en kan bygge applikasjoner for andre enheter og operativsystemer. I figur 4.2.1 så er arkitekturen til applikasjonen illustrert.



Figur 4.2.1 - Illustrasjon av applikasjon arkitektur

Arkitekturen følger design mønsteret model view controller som ofte blir forkortet til mvc. *Model view controller* design mønsteret deler programmet opp i data også kjent som *model*, *view* som er kjent som brukergrensesnittet og en *controller* som hjelper til med kommunikasjon mellom dataen og brukergrensesnittet. Kort oppsummert, *model* holder på data, *view* viser frem data og *controller* flytter på data. I denne applikasjonen er React "view" siden det representerer brukergrensesnittet, Elasticsearch databasen er "model" siden den holder på data som blir vist frem i brukergrensesnittet og Spring-boot er "controller" siden den henter data fra elasticsearch databasen i form av REST requester. REST står for Representational state transfer og er en enkel og effektiv måte å overføre data mellom backend og frontend uavhengig av programmeringsspråk (Visma, 2019).

4.2.2 Databasearkitektur

Sharecat disponerte gruppen med en Elasticsearch database helt i starten av prosjektet. Sharecat krevde at prosjektet skulle ta i bruk elasticsearch. Siden den eksisterende desktop applikasjonen bruker en elasticsearch database så fikk vi utdelt en tilsvarende index.

Elasticsearch er en dokumentorientert database som er optimalisert for søk og gjenfinning av data. Elasticsearch lagrer data i såkalte "indexer", en elasticsearch index er en samling av dokumenter som er i slekt med hverandre. Disse dokumentene er lagret i Javascript object Notation (JSON) format, dokumentene inneholder en mengde med "keys" der hver "key" har en tilsvarende verdi (Elastic, 2021). Dersom vi sammenligner en elasticsearch database med en SQL database så er en index det samme som en SQL tabell og et dokument i indexen er det samme som en rad i en SQL tabell (Figur 4.2.2).

SQL Table

ID	City	Description
1	Bergen	Rainy
2	Alta	Cold

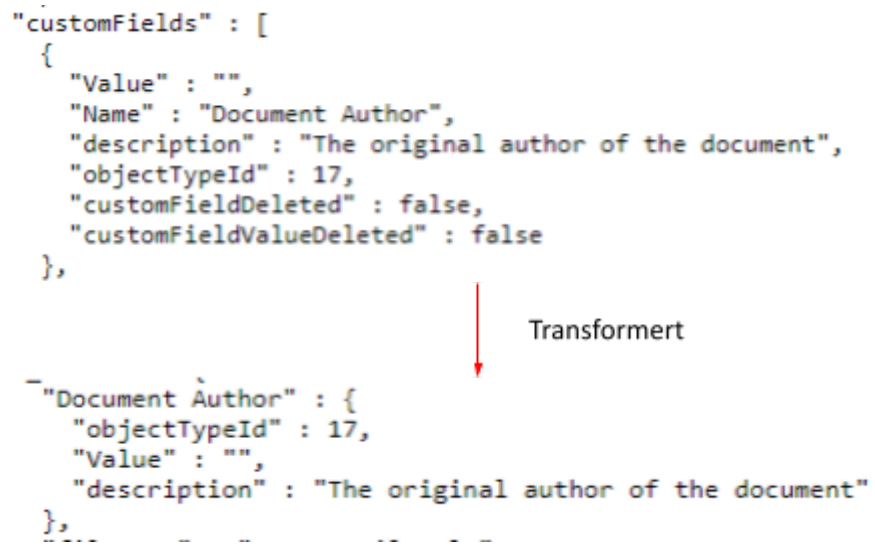
JSON document

```
{
"ID": 1,
"City": "Bergen",
"Description" : "Rainy"
},
{
"ID": 2,
"City": "Alta",
"Description" : "Cold"
}
```

Figur 4.2.2 - Dokument i JSON format sammenlignet med hvordan det hadde sett ut i en SQL tabell

Sharecat sin cis plattformen har veldig store datamengder som skal søkes gjennom og trenger derfor en teknologi som kan raskt finne frem til riktig data når brukeren søker og det er akkurat det elasticsearch er spesielt egnet til. Derfor ble elasticsearch brukt i dette prosjektet som database.

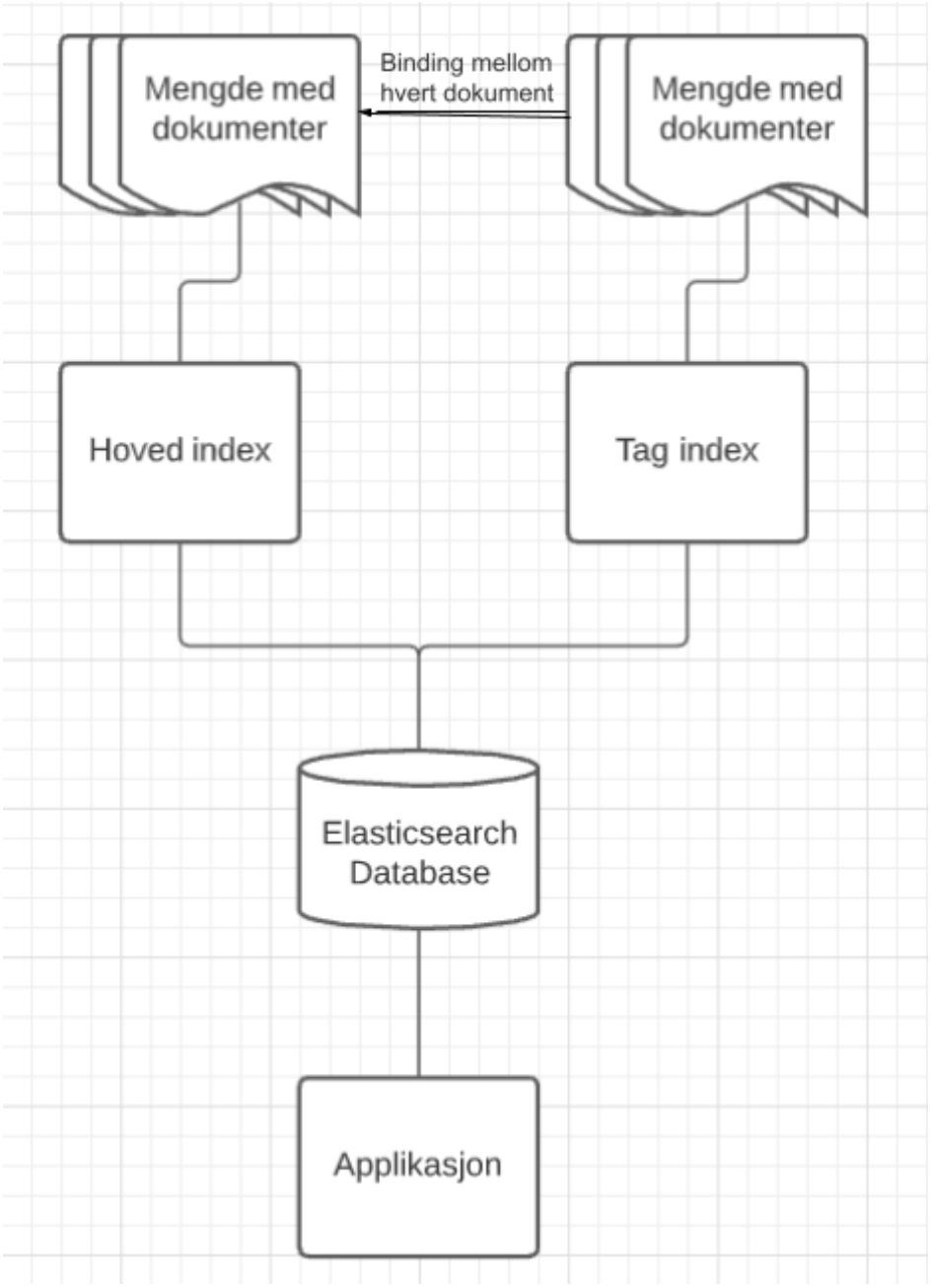
Før gruppen kunne ta i bruk elastic-indexen som ble supplert trengte gruppen å transformere noe data i databasen, en utflating måtte til. Dette måtte gjøres for at dokumentasjonen fra databasen skulle bli mer leselig og brukervennlig når den ble presentert i frontend. Siden elastic-indexen gruppen fikk utdelt var originalt en SQL database som i senere tid ble overført til en dokumentbasert database så inneholdt den en del nøstede verdier. Disse nøstede verdiene måtte bli transformert til objekter, denne transformasjonen kan en se i figur 4.2.3.



Figur 4.2.3 Transformasjon av verdier i database

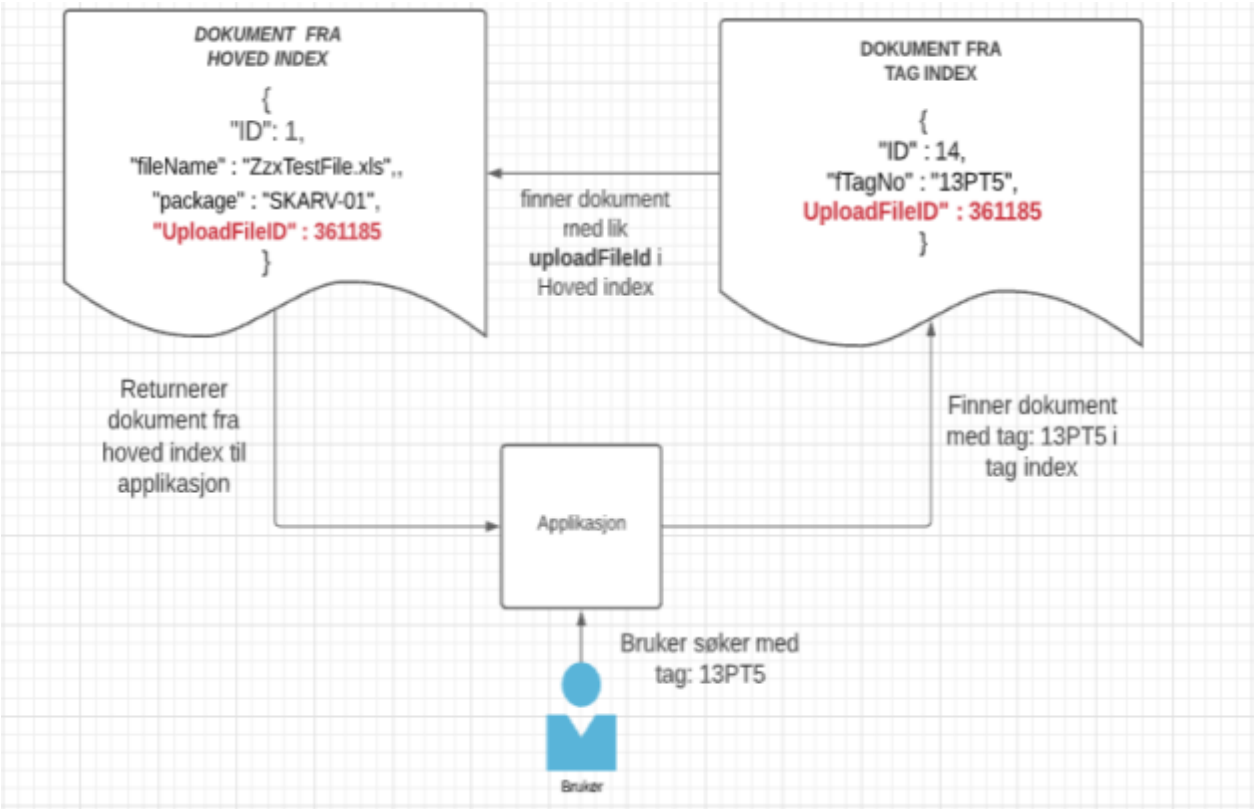
Figuren viser et objekt inne i en nøstet tabell med navn *customFields* som ble transformert til et eget objekt som ikke er inne i den nøstede tabellen. Feltene *customFieldDeleted* og *customFieldValueDeleted* er også fjernet grunnet disse feltene ikke ble brukt og tok unødvendig plass. Feltet, *Value* er tom i den utdelte indexen fra Sharecat, mens andre indexer som sharecat vil bruke med applikasjonen i fremtiden bruker dette feltet, derfor ble den ikke fjernet.

Elasticsearch databaser som ble brukt i applikasjonen består av to indekser. En hoved index som inneholder all dataen som vises i applikasjonen og tag index som inneholder såkalte "tagger" (Figur 4.2.3). Tagger er fysiske identifikatorer som brukere finner på utstyr som de vil finne frem dokumentasjon på.



Figur 4.2.4 - Elasticsearch database arkitektur

Når brukere søker i hoved søkefeltet blir søket utført på hoved-indexen, søket brukeren skriver i feltet blir utført på alle “keys” i indexen. For eksempel, dersom en bruker søker på “test” vil applikasjonen returnere alle dokumenter som har minst en “key” med verdien “test”. Dersom brukere vil søke på tag, trykker brukerne på en knapp i applikasjonen som gir et nytt søkefelt der brukere kun kan søke på tag. Applikasjonen vil da returnere et dokument fra hoved-indexen som er forbundet med taggen som brukeren søkte på. Bindingen mellom hoved index og tag index er illustrert i figur 4.2.4.



Figur 4.2.5 - Illustrasjon som viser forbindelsen mellom tag-Index og Hoved-Index

4.2.3 API forespørsler

Elasticsearch databasen og applikasjonen kommuniserer gjennom et REST api som er kodet i java med Spring-boot som rammeverk. REST forespørsler kommer i fem forskjellige varianter, GET, POST, PUT, DELETE og PATCH. I dette prosjektet ble bare GET forespørsler brukt. Applikasjonen bruker fire GET forespørsler som returnerer enten et helt json objekt eller et key value par (Figur 4.2.5).

Figur 4.2.6 - Json objekt og key value par

GET forespørsel .../view/allDoc.
Returnerer en liste over alle dokumentene i indexen. Denne blir sendt til frontend i form av en liste av JSON-objekter. Frontend transformerer json objektene inn i en tabell slik at de blir mer leselig for brukerne.

GET forespørsel .../view/documents/searchTerm/{term}.
Forespørselen tar i mot en variabel i "{term}" og utfører et søk på dette. Denne returnerer en liste av JSON-objekter korresponderende til søket. Denne requesten er den viktigste i prosjektet ettersom den gir mulighet for brukere å søke opp det dokumentet de trenger.

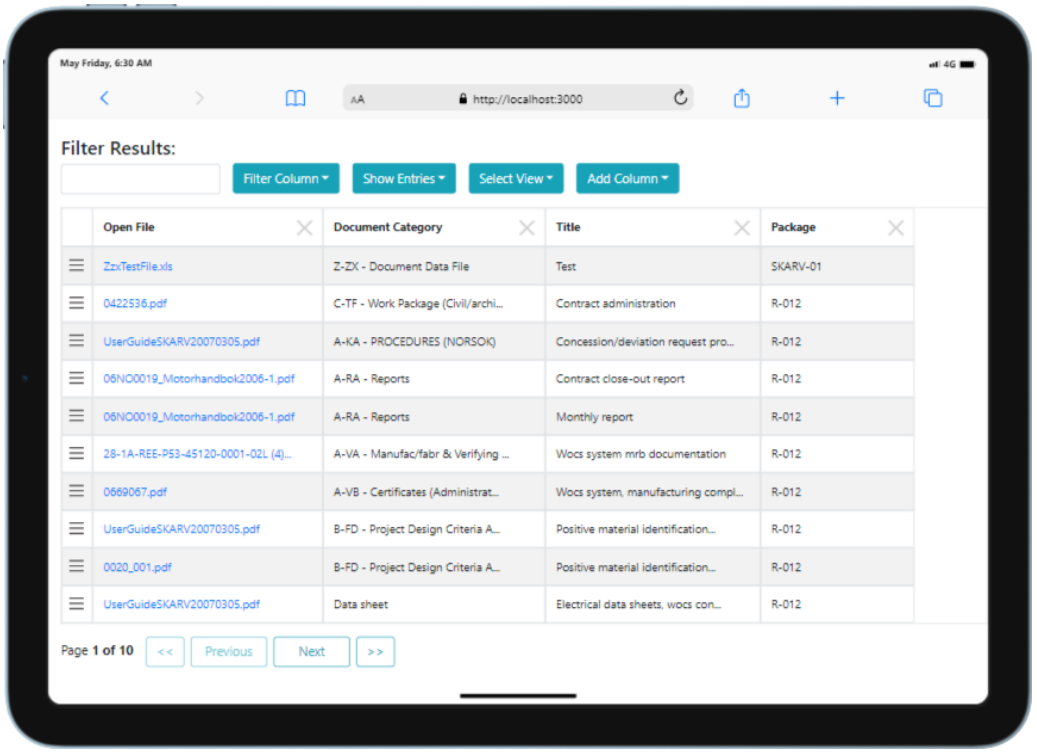
GET forespørsel .../view/documents/uploadID/{uID}.
Denne forespørselen tar imot en variabel i form av en "uploadFileID" og returnerer alle dokumenter som inneholder den korresponderende "uploadFileID". Denne requesten er brukt i sammenheng med den neste for å hente ut dokumenter basert på tag.

GET forespørsel .../view/documents/tagNo/{fldTagNo}.
Forespørselen tar imot en "tagNo" variabel og utfører søk på den, json objektet som returneres inneholder en "uploadFileId" som det igjen blir gjort en forespørsel på for å returnere dokumentet som tilhører den korresponderende "tagNo", slik som illustrert i figur 4.2.5.

4.3 Hvordan applikasjonen fungerer

Etter installasjon av applikasjonen, vil en bruker ha mulighet til å logge inn ved bruk av en føderert Office 365 innlogging.

Etter installasjon av applikasjonen, vil en bruker ha mulighet til å logge inn ved bruk av en føderert Office 365 innlogging. Ved en suksessfull innlogging, vil brukeren få tilgang til alt av innhold i applikasjonen og sendt til dokumentsiden.

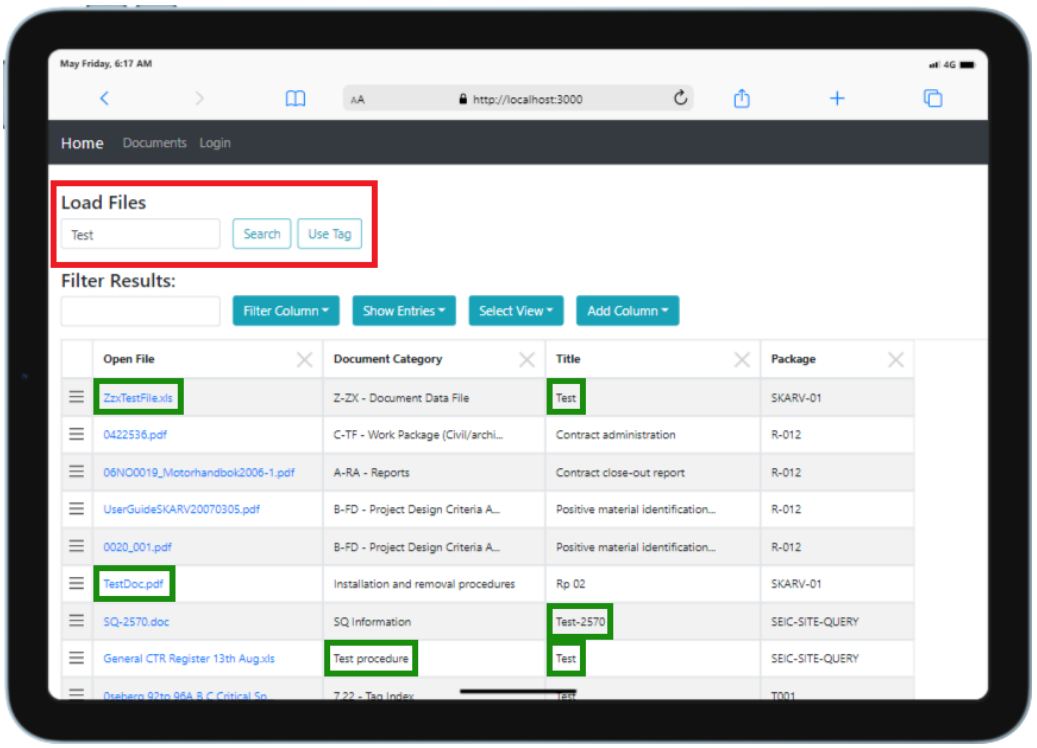


Figur 4.3.1

Figur 4.3.1 viser hva brukeren blir møtt med etter innlogging. Dette er da en side som viser frem data hentet fra Sharecat sin dokumentdatabase i en konfigurerbar tabell. Dette betyr at dataene og kolonnene som vises på figur 4.3.1, er kun et utdrag for å forenkle brukergrensesnittet. Hvordan dette fungerer i praksis, vil forklares når en går over alle funksjoner i neste del.

4.3.1 Henting av data

For å hente ut data har brukeren mulighet til å utføre et søk direkte mot dokumentdatabasen. Dette kan være et søk på attributter (kolonne data), eller basert på en tag. Etter at søkeordet er skrevet inn og brukeren trykker søk, vil den fremviste dataen oppdateres i henhold med søket. For å bruke tag, må brukeren først trykke på «Use Tag». Dette vil returnere alle dokumenter som er koblet opp mot taggen.

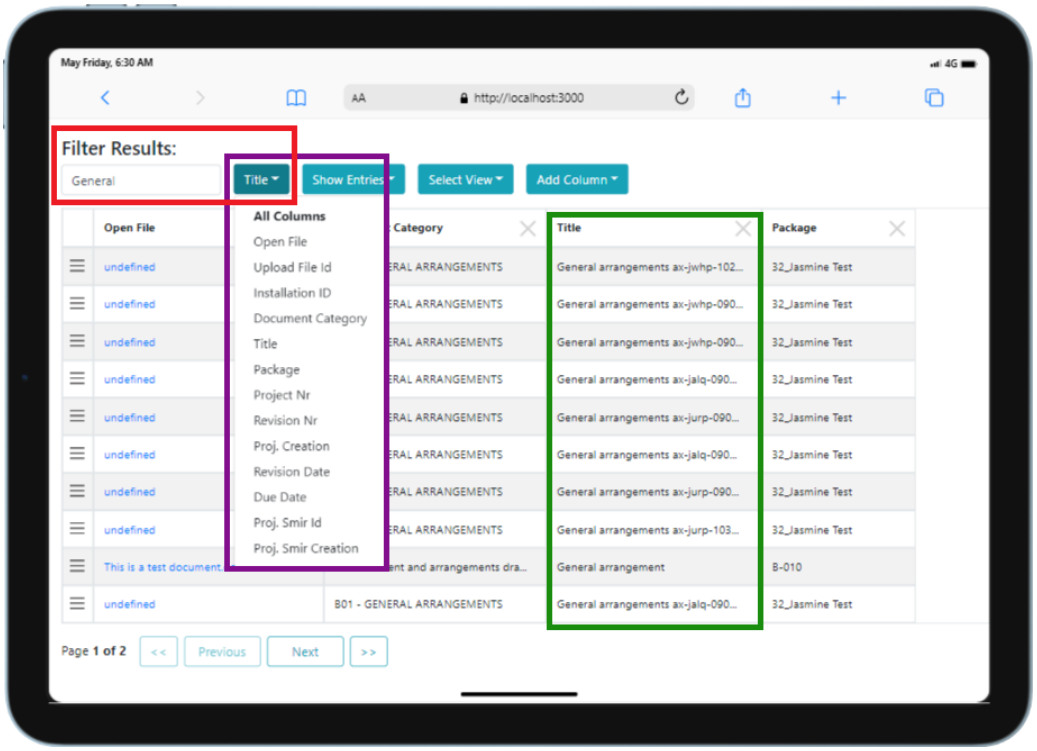


Figur 4.3.2 – Uthenting av data

Figur 4.3.2 viser først markert i rødt hvilke komponenter som inngår i søkefunksjonen. Dette er et skrivefelt, en søkeknapp og en knapp for å veksle mellom å søke på tag eller attributter. I dette eksempelet, er det gjort et søk på attributter med søkeord «Test». Markert i grønt, kan en se hvor ulike attributter matcher søkeordet. Ettersom antall kolonner er kun et utdrag, vil en ikke kunne se en direkte match på alle rader.

4.3.2 Lokalt søk av hentet data

Etter at data har blitt hentet ut, har brukeren mulighet til å gjennomføre et lokalt søk enten på en spesifikk kolonne, eller basert på alle attributter. Dette blir gjort ved bruk av «Filter Results» skrivefeltet og dataene blir oppdatert ett sekund etter brukeren har sluttet å skrive.



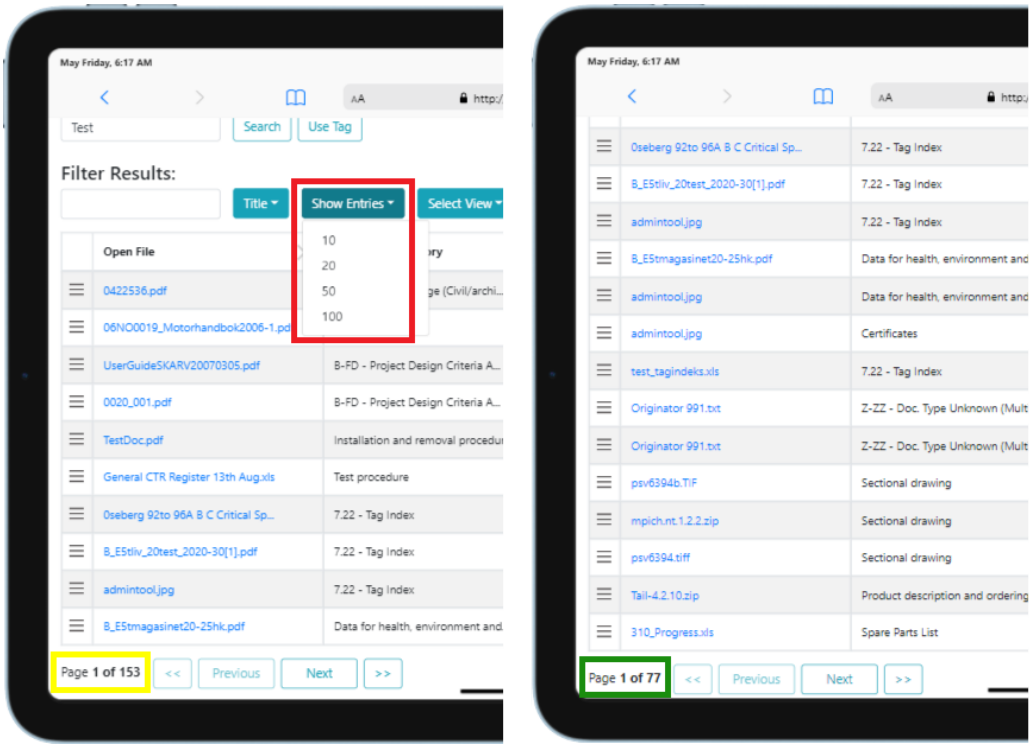
Figur 4.3.3 – Lokalt Filter

Markert i rødt, ser en det lokale filteret basert på søkeordet «General» i «Title» kolonnen. Knappen til høyre, markert både i det rødt og lilla, er hvor brukeren har mulighet til å spesifisere hvilken kolonne søket skal gjøres på. Standardsøket blir gjort på alle kolonner. Markert i grønt, kan en se at alle titler inneholder søkeordet «General» etter at søket har blitt gjennomført.

4.3.3 Endre antall rader

Standardoppsettet, viser kun 10 rader eller instanser per side. Brukeren har deretter mulighet for å bla gjennom resultatene med en meny under tabellen. Ettersom det kan være flere tusen rader, kan brukeren ha ønsker om å endre antall rader som blir vist på hver side. Dette kan bli gjort ved bruk av «Show Entries» knappen, som gir brukeren en dropdown meny med ulike alternativer.

Figur 4.3.4, viser hvordan antall rader og sider endres etter valg av «Show Entries» markert i rødt. I dette eksempelet, ble antall rader endret fra 10 til 20. Markert i gult, kan en se antall sider før endring, som her var 153. Etter endring (markert i grønt) er det kun 77 sider.

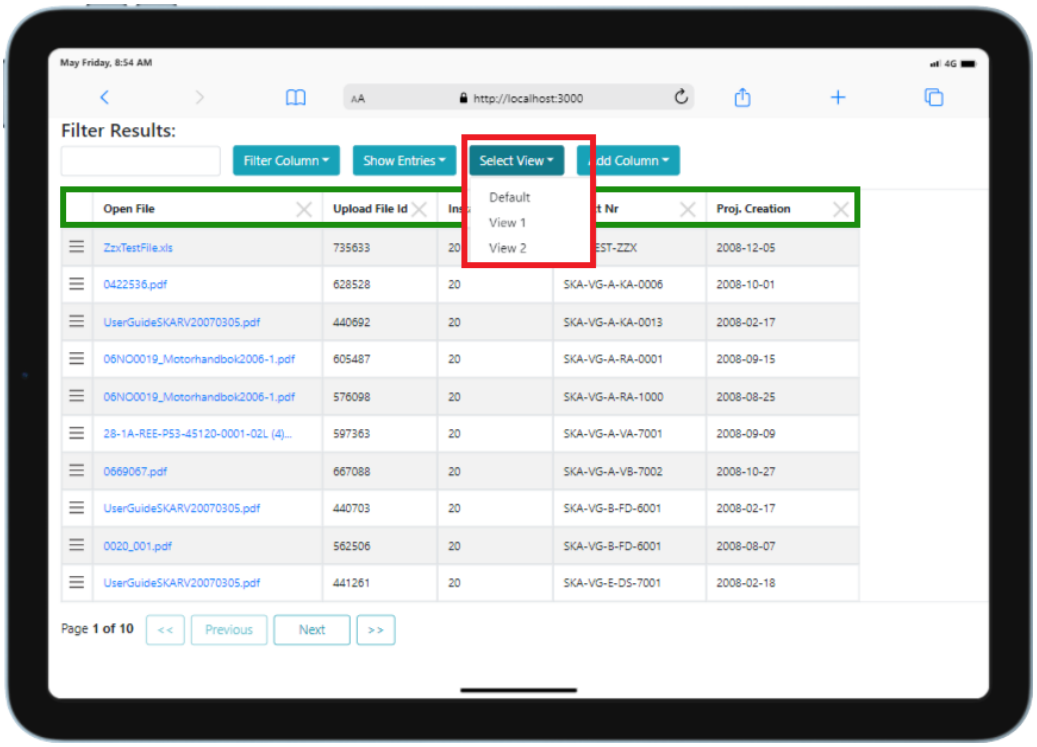


Figur 4.3.4 – Endre antall rader

4.3.4 Valg av kolonneoppsett

Ettersom applikasjonen er i utgangspunktet tenkt som en mobilapplikasjon, er kun noen få kolonner med i standardoppsettet. Det kan allikevel være et behov for å vise frem andre kolonner og en av mulighetene her, er da å velge mellom predefinerte kolonneoppsett som her er bruker betegnelsen «Views».

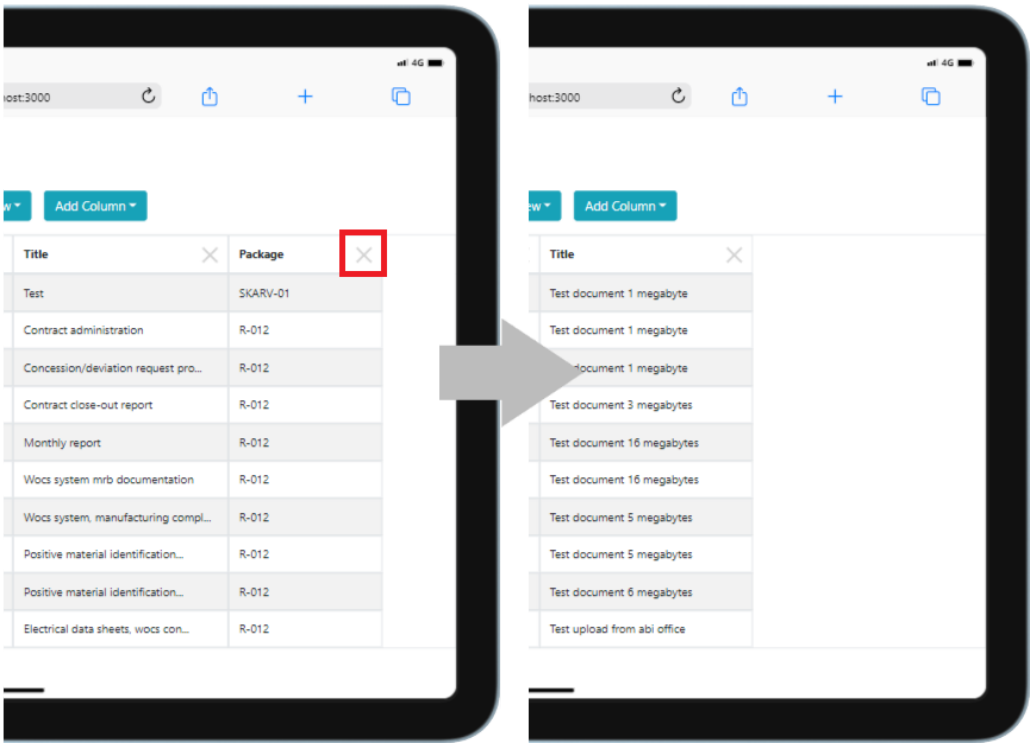
Figur 4.3.5 viser markert i rødt, en dropdown meny «Select View» hvor en kan velge mellom 3 predefinerte oppsett. I dette eksempelet, ble «View 1» valgt og kolonnene ble oppdatert som vist i grønt.



Figur 4.3.5 – Endre kolonneoppsett

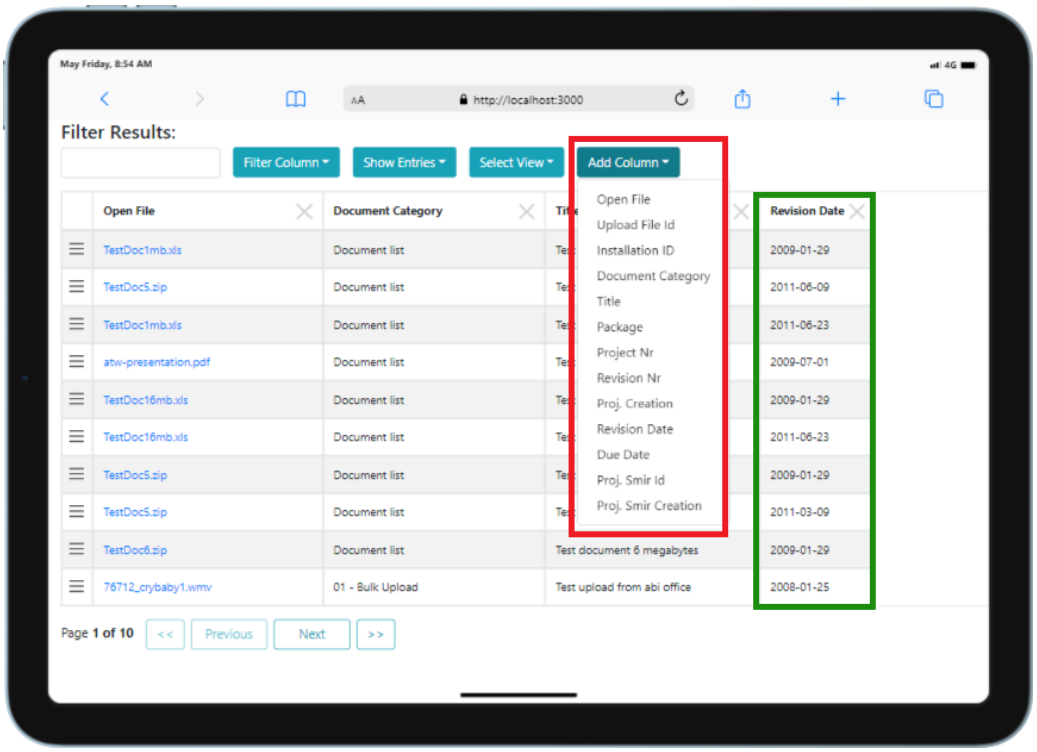
4.3.5 Egendefinert kolonne konfigurasjon

Ettersom de predefinerte kolonne oppsettene kanskje ikke er tilstrekkelig for alle brukere, er det muligheter for å definere individuelle kolonneoppsettet. For å fjerne kolonner, har brukeren mulighet til å trykke på «X» til høyre for hvert kolonnenavn. Dette er markert som rødt i figur 4.3.6.



Figur 4.3.6 – Fjerne kolonne

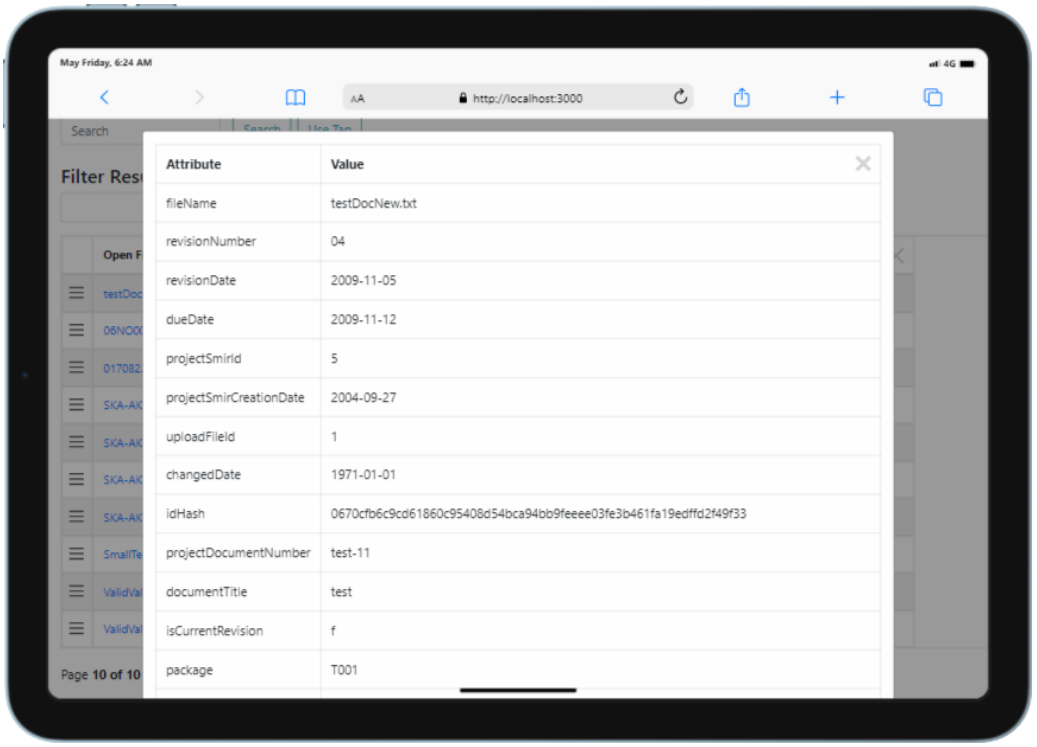
Når en har fjernet kolonner, kan det også være et behov for å legge til kolonner. Dette kan en gjøre ved bruk av «Add Column» dropdown menyen markert i rødt i figur 4.3.7. Brukeren får da mulighet til å velge mellom alle tilgjengelige kolonner og få den lagt til tabellen. I dette eksempelet, ble «Revision Date» lagt til.



Figur 4.3.7 – Legge til kolonne

4.3.6 Se alle attributter

Tabellformatet er bra for å få en oversikt og filtrere data, men ettersom det kun kan være noen få kolonner om gangen, er det et behov for å kunne se mer fullstendig informasjon om instansene. Dette kan gjøres ved å klikke på symbolet helt til venstre i hver rad. Dette vil åpne et vindu med alle attributter og verdier for den instansen. Dette kan bli sett i Figur 4.3.8.

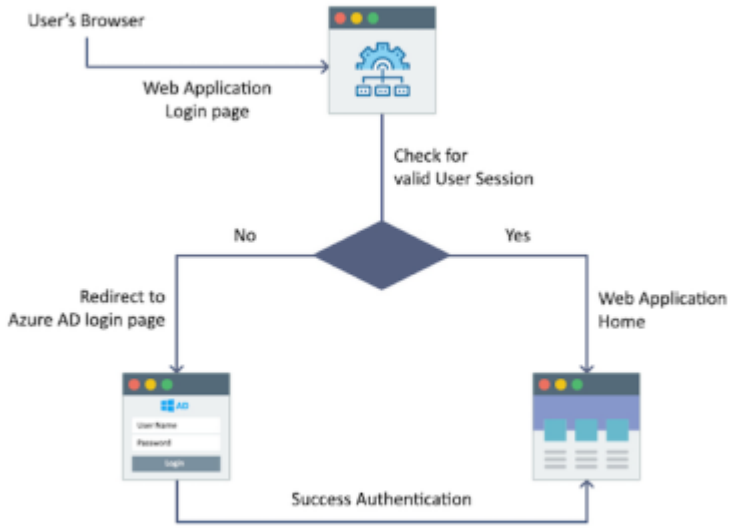


Figur 4.3.8 – Utdypende informasjon

4.4 PWA og Microsoft føderering

Gruppen valgte å lage applikasjonen som en Progressive Web App(PWA). En PWA innebærer en applikasjonsprogramvare levert via nettet. Dette er vanligvis bygd av vanlige webteknologier, inkludert HTML, CSS og Javascript. Poenget med en PWA er å ha en applikasjon som fungerer på alle plattform, som benytter seg av en nettleser. Enten dette er mobilt (eks: Pad, Mobil) eller stasjonær (eks:PC). En annen fordel er bruken av service workers. En relevant situasjon som ble gjort åpenbart er at tilgang til internett kan være vanskelig for brukere. Dette fordi feltarbeidere er i områder med tykke vegger som ikke fører wifi-signaler godt, og fordi wifi er pålagt skrudd av i hensyn til sikkerhetsmessige årsaker. Det ble tatt grep på ved hjelp av service-workers. Service-workers tillater oss å cache data og deretter hente det ut i offline-modus, som gjør at applikasjonen kan fungere uten wifi.

Identity access management(IAM) har vært en stor del av prosjektet. På applikasjonen skjer dette gjennom en Azure SSO setup slik som figur 4.4.1 illustrerer.



Figur 4.4.1 Azure SSO

Her er det slik at Azure tar seg hånd om alt av login logikk. Slik at brukere trenger kun en login for å få tilgang til denne og alle andre applikasjoner satt opp mot Microsoft. Videre har firmaer også mulighet for å kunne delegere tilgang, holde styr på samtykke og kreve ekstra autentisering som Multi Factor Authentication(MFA) blant ansatte.

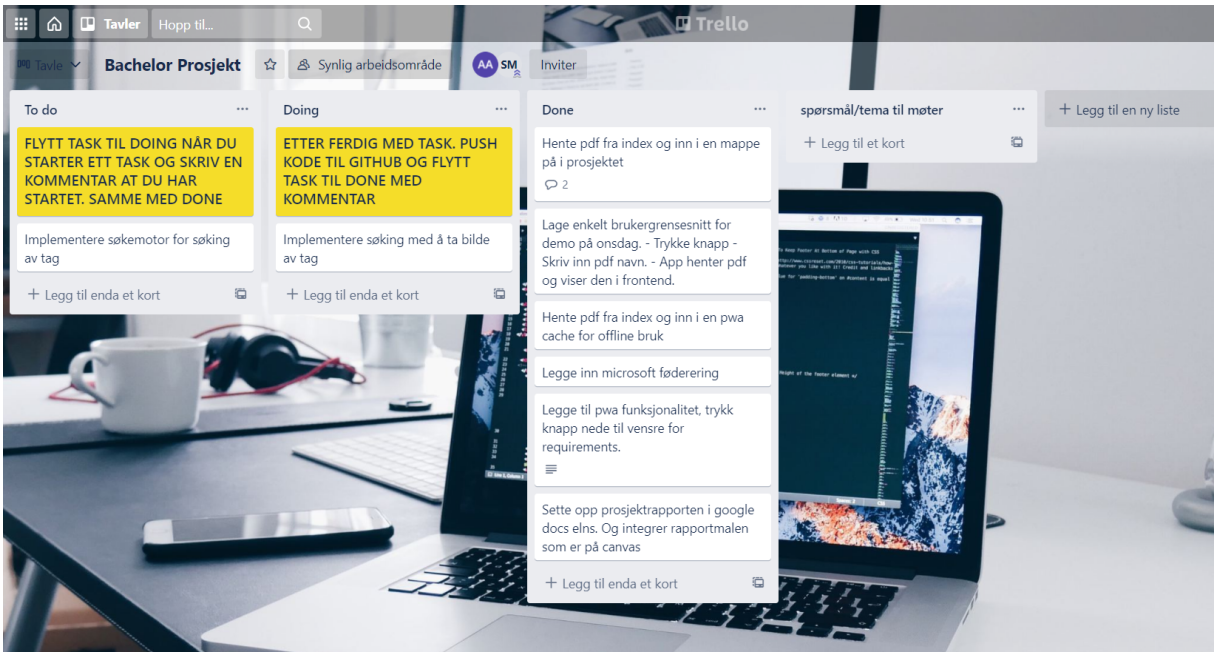
5 Evaluering

5.1 Evalueringsmetode

Prosjektet ble preget med tilbakemeldinger og diskusjon angående løsninger til forskjellige oppgaver. Dette pågikk kontinuerlig siden prosjektstart og kan sies å ha ført prosjektet mye lengre enn først antatt. Nedenfor ser man de forskjellige evalueringspunktene.

5.1.1 Utviklingsmetode

Gruppen benyttet seg av Scrumban som utviklingsmetodikk. Scrumban som nevnt i 3.4.1 er en metodikk som kombinere Scrum og Kanban. Oppsettet av Scrumban i prosjektet var ved bruk av et Trello brett.



Figur 5.1.1: her er brettet delt opp i fire kolonner: ”To do”, “Doing”, “Done” og “spørsmål/tema til møter”.

Effekten av dette var liten til å begynne med siden prosjektet i seg selv hadde ingen klar vei i form av arbeid og mål man hadde lyst til å oppnå. Ved kommunikasjon med både veileder, prosjekteier og innad i gruppen ble dette redegjort for. Her ble løsning å lage “vertical slices”. Nemlig, en rød tråd gjennom hele applikasjonen, som er lett å oppdage av brukere. Noen eksempler på dette er å kunne se data, logge inn eller bruke søkemotoren.

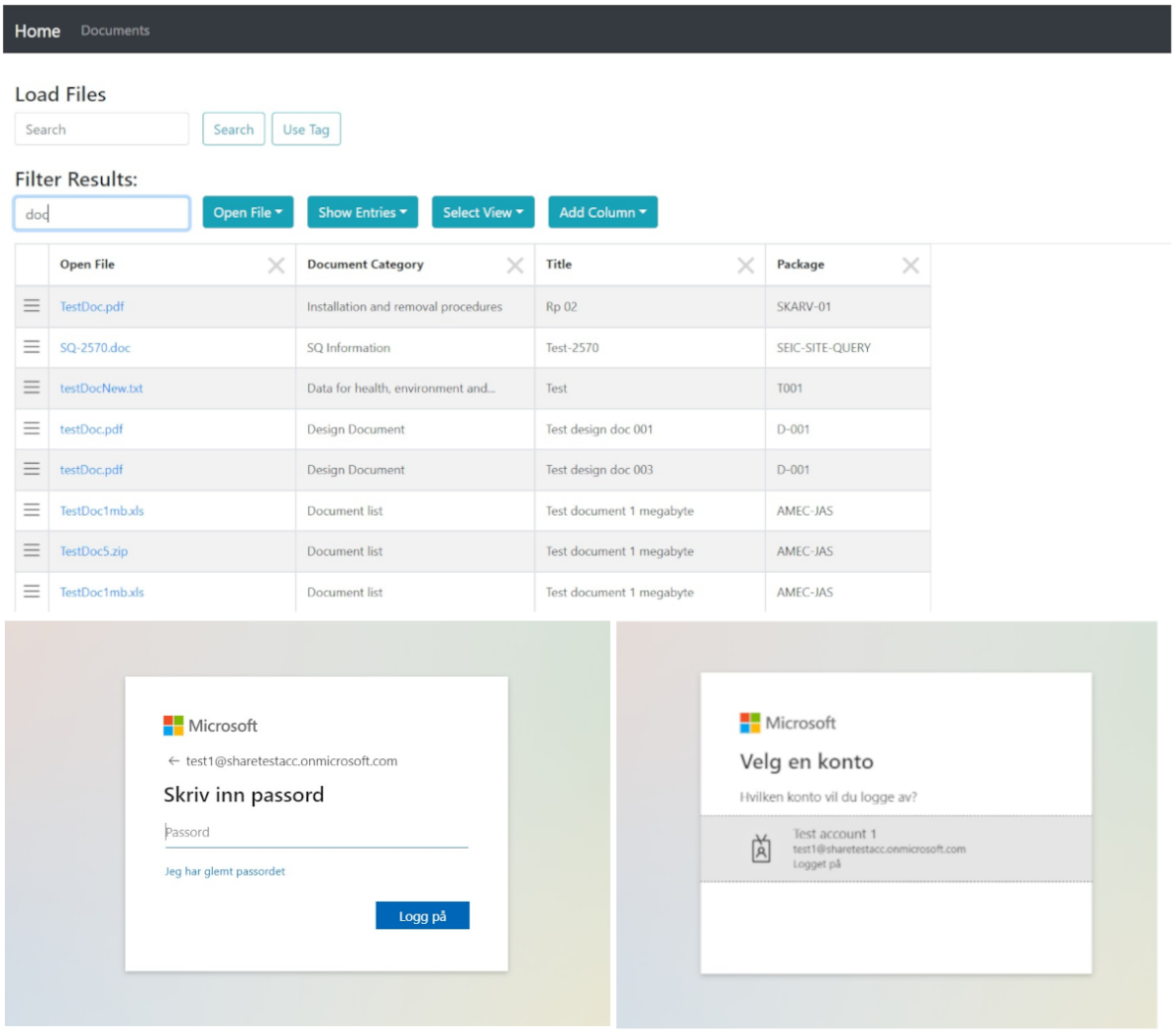
For å evaluere PWA brukes en enkelt side. Siden skal gjøre tilgjengelig login, søkemotor og filer for brukere. Brukere ved login side få mulighet til å installere web applikasjonen. På mobile enheter skal icon-format og åpning være identisk til andre apper ved installasjon.

Videre testes login ved hjelp av organisasjons-kontoer. Her kan enhver konto logge inn før det ble redirected til egen organisasjons office 365 login. Dette innebærer at kontoen har blitt klarert for tilgang av organisasjonen policy på forhånd.

Figur 5.1.1: Her viser policy at brukeren har ikke blitt klarert for tilgang til applikasjon

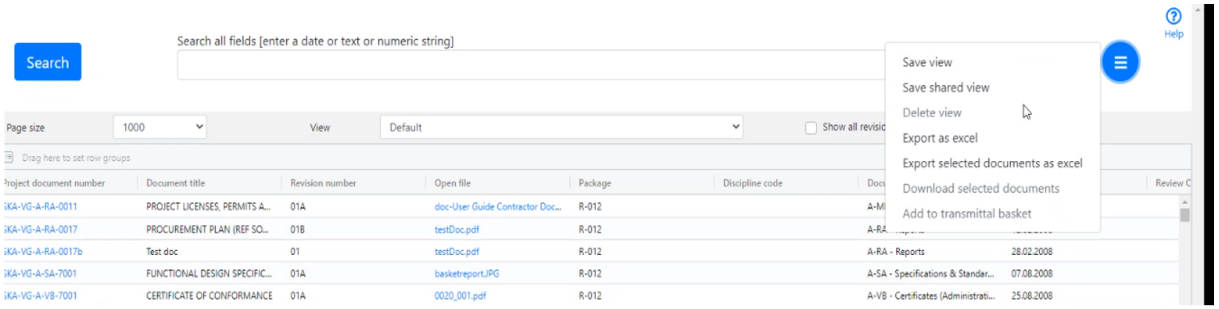
5.2 Evalueringsresultat

På grunn av Covid-situasjonen har det vært begrensninger for evaluering av produktet. Dette kommer spesielt fram ved mangel på testing blant brukere. De viktigste funksjonene på plass, og det er produkteiere fornøyd med. Applikasjonen har nådd mål med søkemotor som har mulighet for filtrering også basert på tag. Videre er applikasjonen satt opp med en Identity Access Managment (IAM) løsning. Noe som gjør login integration med bedrifter mye enklere.



Figur 5.2.1: Her vises IAM login/logout og søkemotor.

Dette resultatet kan sammenlignes med den gamle søkemotoren.



Figur 5.2.2: Gammel søkeløsning

Her kan man se at den mest åpenbare forbedringen er formaterings løsningen for forskjellige skjermer. Dette henvises til i figur 4.3.5 ved valg av kolonneoppsett og figur 4.3.6 ved laging av egendefinert kolonneoppsett. Grunnlaget til at man har to oppsett her er for å gi brukere mer frihet, og for å unngå api-forstyrrende oppførsel ved krav av oppsett-endring.

6 Diskusjon

6.1 Valg av løsning

I dette prosjektet ble valg av løsning veldig påvirket av oppdragsgiver allerede før prosjektet startet. De hadde veldig sterke ønsker om både rammeverk og programmeringsspråk som skulle brukes. Applikasjonen skulle være en PWA (Progressive web app), som krever en backend for underliggende logikk og en frontend for å vise frem applikasjonen til klienter. Dette gjorde at gruppen hadde begrenset med valg rundt applikasjonens utforming, men det en kan se nærmere på et hvordan dette påvirket applikasjonen og utviklingen.

6.1.1 Programmeringsspråk og rammeverk

Selv om oppdragsgiver i utgangspunktet hadde ønsker om at backend skulle bruke .NET Core, valgte gruppen å gå for Java som alle hadde mer erfaring med. Underveis i prosjektet, viste det seg at dette var det rette valget og ta ettersom prosjektet fort ble mer omfattende enn først antatt. Spring Boot ble også valgt som rammeverket rundt backenden noe som i dette prosjektet fungerte ganske bra. Det var aldri noen store komplikasjoner eller restriksjoner og selve utviklingen gikk ganske greit.

For frontend, var det eneste valget å ta hvilke rammeverk en skulle bruke siden programmeringsspråk (HTML, JS og CSS) allerede var bestemt når en valgte å gå for en PWA. For rammeverk, valgte gruppen å gå for React som flere av gruppemedlemmene allerede hadde erfaring med. Dette er en komponentbasert utvikling hvor en lager biter av applikasjonen og kombinerer disse for å lage komplekse nettsider. Utviklingen gikk i starten veldig greit, en bygget opp applikasjonen og funksjoner etter hver iterasjon, men når en nærmet seg slutten av applikasjonen, kom det frem at det var restriksjoner for filhåndtering og fremvisning av ulike filtyper som gjorde utviklingen mye mer krevende.

6.1.2 Brukergrensesnitt

Under frontend, ligger også brukergrensesnitt og utforming av applikasjonen. Her valgte gruppen tidlig å bruke Bootstrap som har predefinerte CSS og JS filer for animasjon og styling. Dette gjorde at gruppen ikke trengte å tenke for mye på selve designet av komponenter, men mer om helheten av applikasjonen. Det var aldri et klart design på applikasjonen, men det ble mer at ettersom flere funksjoner ble lagt til, ble også designet sakte endret for å oppfylle kravene. Dette gjorde at et design som så greit ut halvveis i prosjektet, fort begynte å bli litt rotete i slutten når flere og flere funksjoner hadde blitt lagt til. Det hadde vært mulig å designe applikasjonen på nytt, men ettersom det allerede var tidsbegrensninger, ble dette ikke prioritert.

6.1.3 Applikasjonens funksjoner

Gjennom hele prosjektet, var det god dialog med oppdragsgiver hvor de forklarte hvilke funksjoner de ønsket og gruppen prøvde å oppfylle deres ønsker så godt som mulig. Dette fungerte fint i starten på prosjektet hvor det var mer fundamentale funksjoner som ble utviklet, men rundt halvveis i prosjektet, begynte det å bli en mismatch mellom hva de ønsket og hva gruppen kunne levere på en iterasjon. Dette gjorde at i stedet for å gjøre en god jobb på en funksjon, ble det gjort en ok jobb på flere, hvor en hadde funksjoner fra tidligere iterasjoner som måtte fikses på etter de egentlig skulle vært ferdige. Oppdragsgiver begynte også å komme med flere og flere ikke essensielle funksjoner som de ønsket som gikk på bekostning av hovedfunksjonene. Her burde nok gruppen ha gitt en klarere beskjed om hva som var oppnåelig på en iterasjon og heller fokusert på hovedfunksjonene først selv om oppdragsgiver også hadde andre ønsker.

7 Konklusjon og videre arbeid

Hvor bra kan teknisk data og dokumentasjon fra CIS plattformen til Sharecat tilgjengeliggjøres for brukere på mobile løsninger i form av en mobilapplikasjon.

I prosjektet ble det utforsket hvor bra gruppen klarte å utvikle en slik applikasjon. Sharecat har allerede en eksisterende desktop applikasjon som ikke fungerer bra i felt. Gruppen hadde ikke tilgang til denne desktop applikasjonen som gjorde at prosjektet ble bygget fra bunn.

7.1 Oppsummering av mål

Hovedmålet med prosjektet var å lage en mobilapplikasjon som gir tilgang til teknisk data og dokumentasjon fra sharecat sin CIS plattform med et brukergrensesnitt egnet for mobile enheter. Andre delmål inkluderte.

- Transformere data i elasticsearch database slik at dataen kan presenteres i frontend på en leselig måte.
- Logg inn system
- Makinlæringsmodell for bildeprosessering av tekst slik at brukere kan gjøre søk gjennom å ta bilde av en streng med bokstaver kalt “Tag”.
- Å ha mulighet til å lagre dokumentasjon på enheten slik at brukere kan lagre dokumentasjon for offline bruk.

7.2 Oppnådde mål

Gruppen utviklet en applikasjon der brukere kan hente ut teknisk data og dokumentasjon fra CIS plattformen til sharecat. Applikasjonen ble utviklet som en PWA (Progressive web app), inkluderer et Logg inn system som tar i bruk Azure SSO og er koblet opp mot en Elasticsearch database. Applikasjonen har også søkefunksjonalitet og gir brukere mulighet til å konfigurere hvordan dokumentasjonen skal presenteres.

Ettersom uthenting av dokumentasjon og teknisk data ble implementert i applikasjonen så kan en konkludere med at hovedmålet ble oppnådd. Applikasjonen har også et logg inn system med hjelp av Office 365 Logg inn. Dataen i elasticsearch databasen ble også transformert. Det som gjensto etter innlevering av rapport var lagring av dokumentasjon lokalt og en maskinlæringsmodell for bildeprosessering. Hovedgrunnen til at disse delmålene ikke ble nådd er tid. Gruppen brukte mye tid på å sette seg inn i prosjektet og transformeringen av data i databasen, dette gjorde at starten av utviklingen av prosjektet tok litt lengre tid en først antatt.

Spørsmålet som gjenstår er hvor bra mobilapplikasjonen ble, og om den er brukervennlig for målgruppen. En god måte å finne dette ut på hadde vært å teste applikasjonen opp mot målgruppen som er arbeidere i felt på oljeplattformer. Ettersom ingen bruker testing ble utført er det vanskelig å svare på akkurat dette.

7.3 Videre arbeid

Videre arbeid med prosjektet kan innebære å implementere mer funksjonalitet og gjennomføre brukertesting.

7.3.1 Implementering av funksjonalitet

Dersom applikasjonen skulle oppnådd alle målene manglet det fortsatt noe funksjonalitet. Dersom gruppen hadde hatt mer tid så ville det blitt lagt til offline funksjonalitet med å lagre dokumentasjon lokalt, en maskinlæringsmodell for bildeprosessering var også en funksjonalitet som gruppen kunne implementert. Dersom gruppen kunne utføre brukertesting så kunne en også finne ut om det er mer funksjonalitet som eventuelt mangler.

7.3.2 Brukertesting

Brukertesting er en veldig nyttig måte å finne ut hva som fungerer bra og hva som ikke fungerer bra med applikasjonen. En brukertest gir en innsikt i nøyaktig hva som er utfordrende for brukeren, og man får samtidig mulighet til å øke kvaliteten på det ferdige produktet (guilty, 2019). Dersom gruppen kunne utført brukertesting så ville en kunne gjort forbedringer basert på tilbakemeldinger. Dette hadde forbedret applikasjonen og gitt målgruppen det de trenger. På denne måten så hadde applikasjonen hatt høy sannsynlighet for suksess hos sluttbrukerene.

8 Referanser og Litteratur

8.1 Referanser i rapport

Visma (2019) Hva er Api? Og 9 andre spørsmål og svar som api [Internett]. Tilgjengelig fra https://www.visma.no/blogg/hva-er-api-sporsmal-og-svar/ (Hentet 15. Mai 2021)

Elastic (2021) What is elasticsearch [Internett]. Tilgjengelig fra https://www.elastic.co/what-is/elasticsearch (Hentet 16.Mai 2021)

Guilty (2019) Brukertesting av nettside - hvorfor er det viktig? [Internett]. Tilgjengelig fra https://guilty.no/blogg/hvorfor-brukertesting-av-nettside-er-viktig (Hentet 31.Mai 2021)

Linkedin (2021) Sharecat solutions, om oss [Internett]. Tilgjengelig fra https://no.linkedin.com/company/sharecat-solutions-as (Hentet 2.Juni 2021)

Planreview (2021) What is Scruman? [Internett]. Tilgjengelig fra https://www.planview.com/no/resources/guide/what-is-scrum/lkdc-what-is-scrumban/ (hentet 16. Mai 2021)

8.2 Litteratur brukt i prosjekt

React (2021) React dokumentasjon [Internett]. Tilgjengelig fra https://reactjs.org/docs/getting-started.html

Elasticsearch (2021) Elastic stack and product documentation [Internett]. Tilgjengelig fra https://www.elastic.co/guide/index.html

Web.dev (2021) Progresive Web Apps [Internett]. Tilgjengelig fra https://web.dev/progressive-web-apps/

Spring (2021) Spring Boot Reference Documentation [Internett]. Tilgjengelig fra https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/

React-table (2021) Lightweight and extensible data tables for React [Internett] Tilgjengelig fra https://react-table.tanstack.com/

Docker (2021) Docker dokumentasjon [Internett]. Tilgjengelig fra https://docs.docker.com/

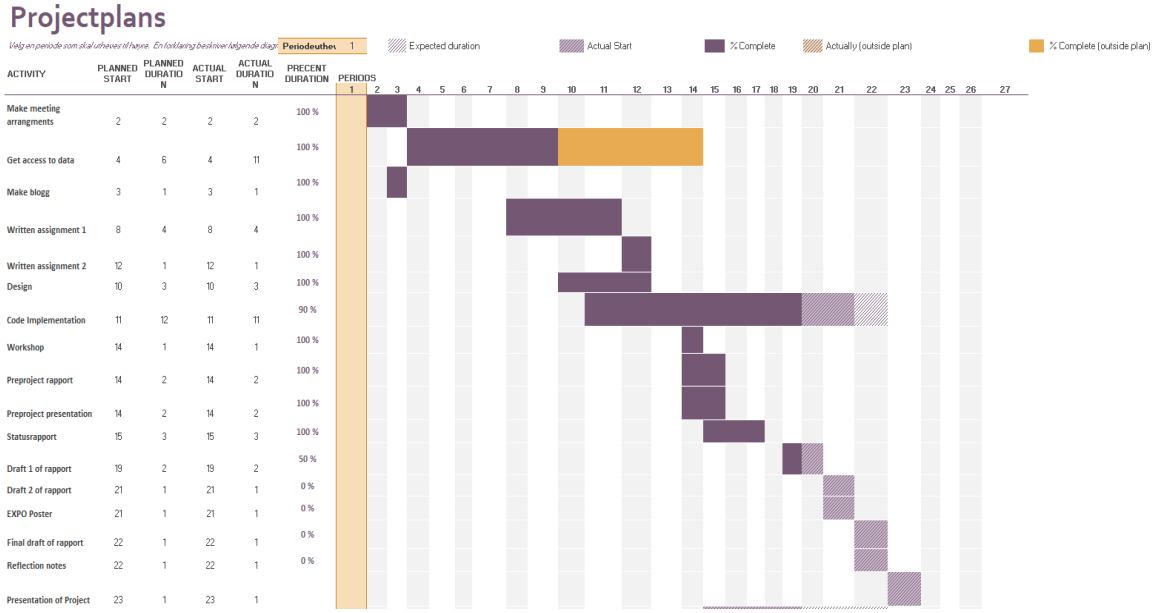
9 Appendix

9.1 Risiko Liste

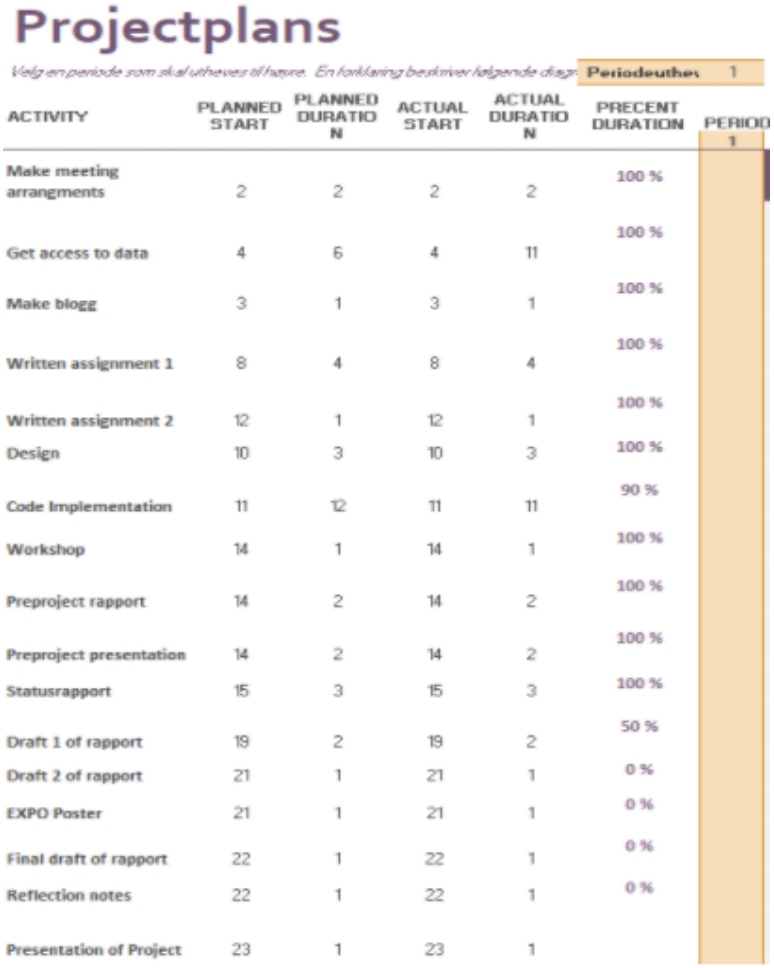
Risikofaktor	S	K	RF	Tiltak	Interessent	Fase
Problemer med å forstå prosjekt oppgaven	5	9	45	Stille presise spørsmål om prosjektet til oppdragsgiver	Gruppen og oppdragsgiver	Utviklingsfasen
Feil Kommunikasjon mellom oppdragsgiver og gruppen	9	3	27	Regelmessige møter hvor en diskuterer fremgangsmåter og prosjektet	Gruppen og oppdragsgiver	Utviklingsfasen
Korrupte filer / mister data	3	7	21	Passe på å ha backup av data	Gruppen, oppdragsgiver og sluttbrukere	Utviklingsfasen
Problemer grunnet digitalt samarbeid	6	3	18	Ha alternative kommunikasjonskanaler	Gruppen og oppdragsgiver	Kontinuerlig
Uenigheter i gruppen skaper forsinkelser	3	2	6	Ha en bestemmelsesprosess om konflikter skulle oppstå	Gruppen og oppdragsgiver	Kontinuerlig
Uferdig / Ikke brukbar applikasjon	3	9	27	God planlegging og holde prosjektplanen	Oppdragsgiver og sluttbrukere	Sluttfasen
Sykdom i gruppen / nøkkelpersoner	5	5	25	Lage en prosjektplan som tar høyde for noen forsinkelser slik at en ikke får for lite tid.	Gruppen og oppdragsgiver	Kontinuerlig
Mangel på brukertesting av applikasjon	7	5	35	Finne brukere som er villig å teste applikasjonen, helst brukere målgruppen	Gruppen og oppdragsgiver	Sluttfasen
Applikasjon ikke kompatibel med nok enheter	6	7	42	Teste applikasjonen på forskjellige enheter	Gruppen, oppdragsgiver og sluttbrukere	Sluttfasen

S: Sannsynlighet K: konsekvens RF: Risikofaktor

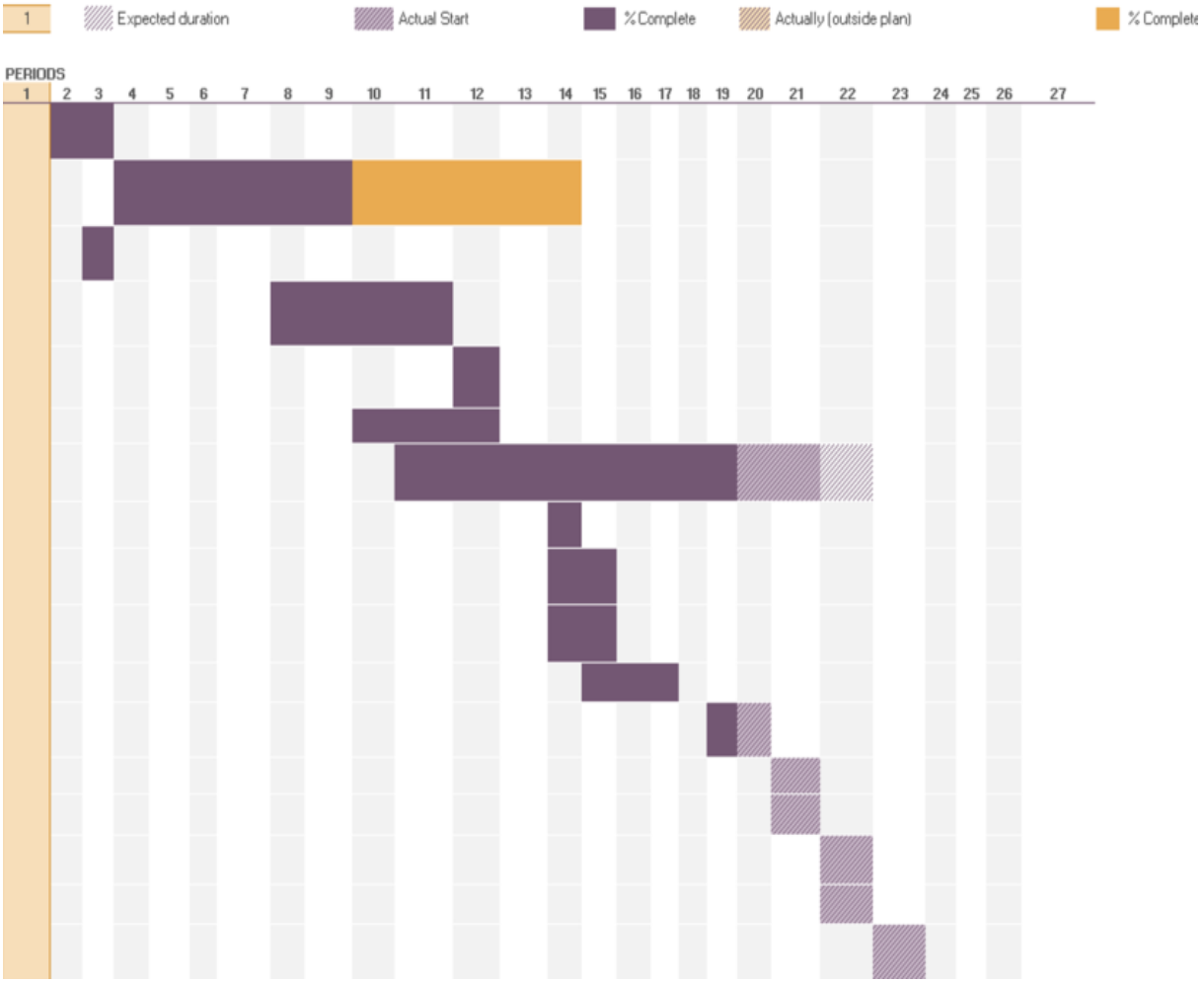
9.2 Gant diagram



Figur 9.2.1 Gant diagram komplett



Figur 9.2.2 Gant diagram forstørret del 1 av 2



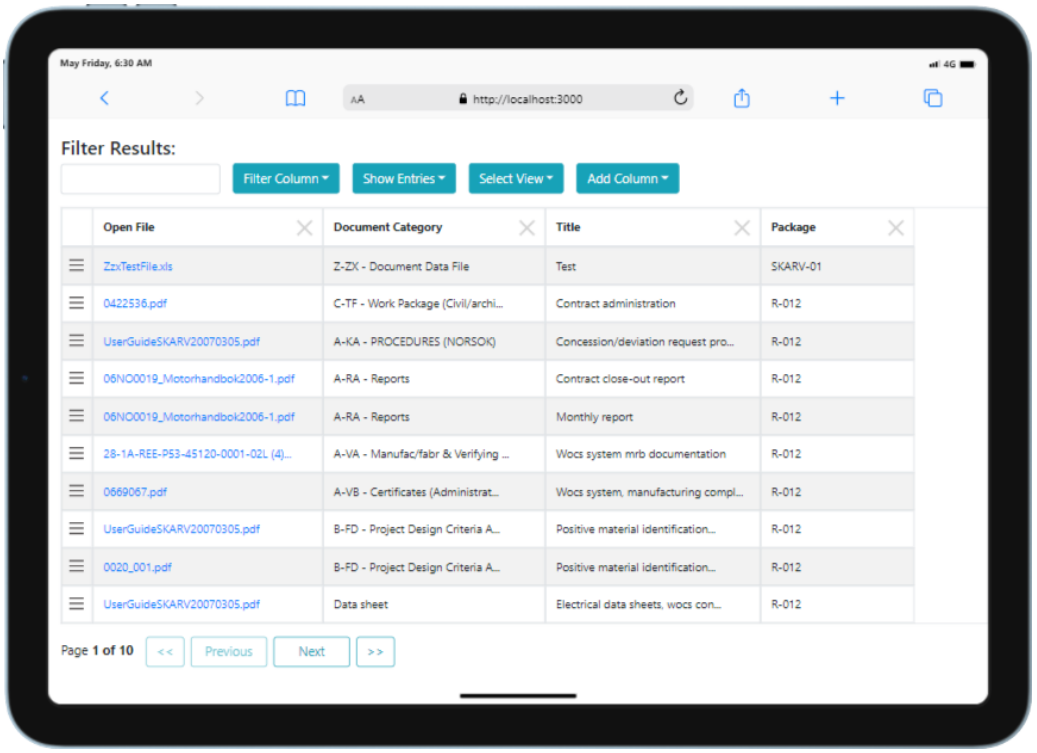
Figur 9.2.2 Gant diagram forstørret del 2 av 2

9.3 Manual

Hvordan applikasjonen fungerer

Etter installasjon av applikasjonen, vil en bruker ha mulighet til å logge inn ved bruk av en føderert Office 365 innlogging.

Etter installasjon av applikasjonen, vil en bruker ha mulighet til å logge inn ved bruk av en føderert Office 365 innlogging. Ved en suksessfull innlogging, vil brukeren få tilgang til alt av innhold i applikasjonen og sendt til dokumentsiden.

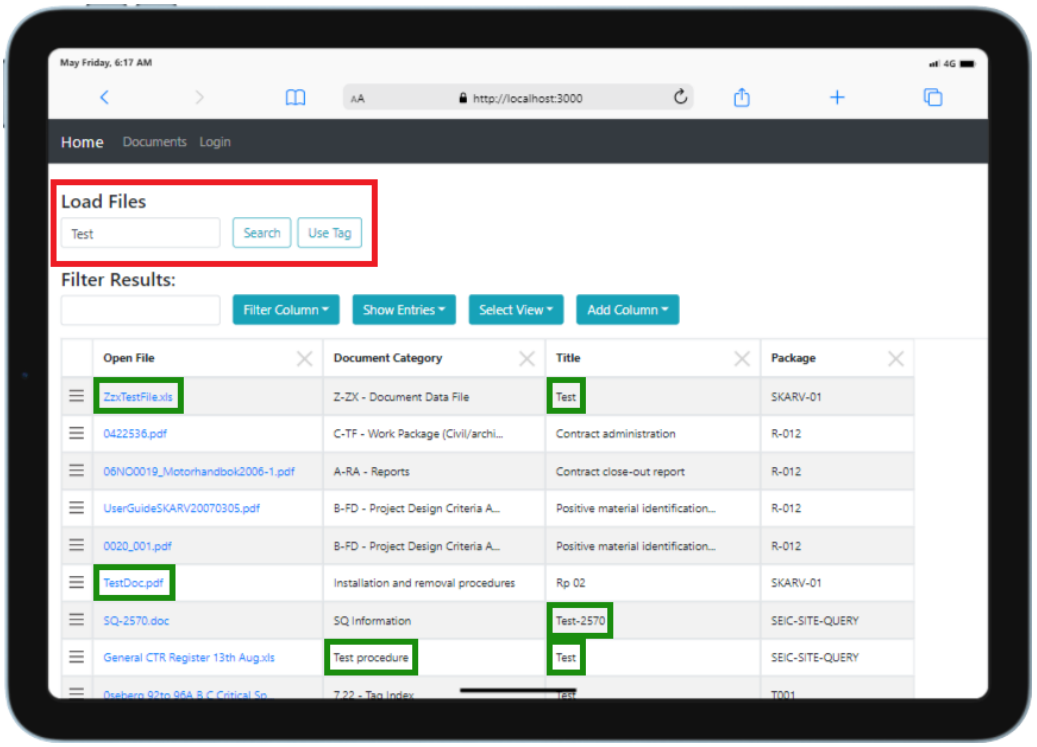


Figur 4.3.1

Figur 4.3.1 viser hva brukeren blir møtt med etter innlogging. Dette er da en side som viser frem data hentet fra Sharecat sin dokumentdatabase i en konfigurerbar tabell. Dette betyr at dataene og kolonnene som vises på figur 4.3.1, er kun et utdrag for å forenkle brukergrensesnittet. Hvordan dette fungerer i praksis, vil forklares når en går over alle funksjoner i neste del.

Henting av data

For å hente ut data har brukeren mulighet til å utføre et søk direkte mot dokumentdatabasen. Dette kan være et søk på attributter (kolonne data), eller basert på en tag. Etter at søkeordet er skrevet inn og brukeren trykker søk, vil den fremviste dataen oppdateres i henhold med søket. For å bruke tag, må brukeren først trykke på «Use Tag». Dette vil returnere alle dokumenter som er koblet opp mot taggen.

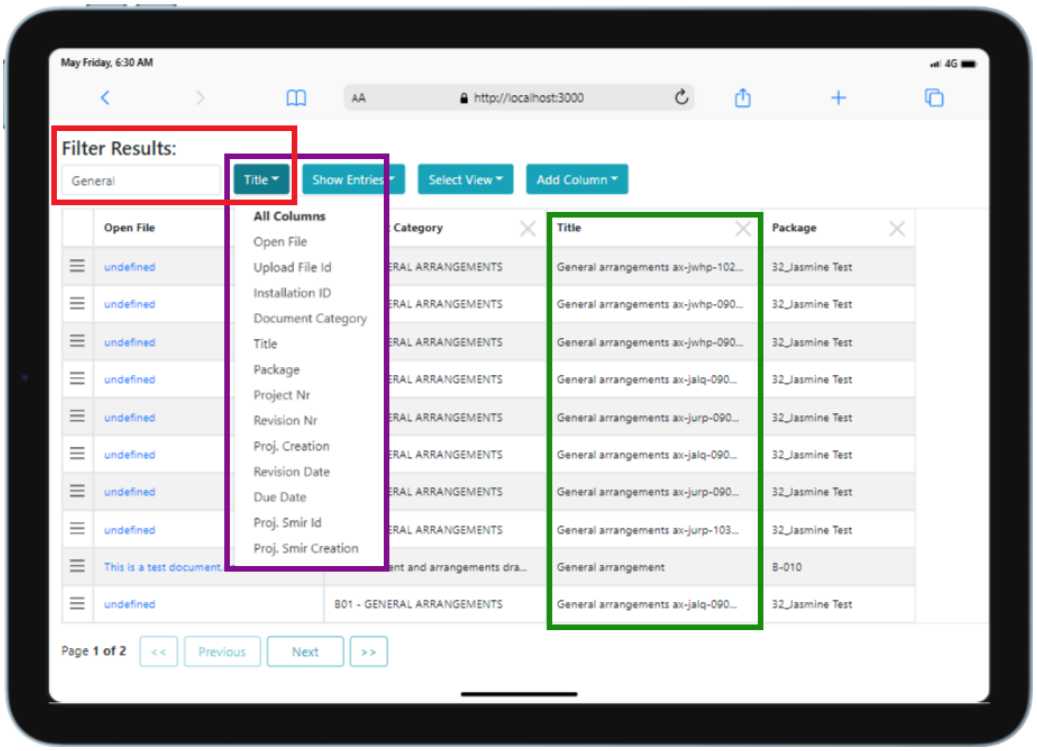


Figur 4.3.2 – Uthenting av data

Figur 4.3.2 viser først markert i rødt hvilke komponenter som inngår i søkefunksjonen. Dette er et skrivefelt, en søkeknapp og en knapp for å veksle mellom å søke på tag eller attributter. I dette eksempelet, er det gjort et søk på attributter med søkeord «Test». Markert i grønt, kan en se hvor ulike attributter matcher søkeordet. Ettersom antall kolonner er kun et utdrag, vil en ikke kunne se en direkte match på alle rader.

Lokalt søk av hentet data

Etter at data har blitt hentet ut, har brukeren mulighet til å gjennomføre et lokalt søk enten på en spesifikk kolonne, eller basert på alle attributter. Dette blir gjort ved bruk av «Filter Results» skrivefeltet og dataene blir oppdatert et sekund etter brukeren har sluttet å skrive.



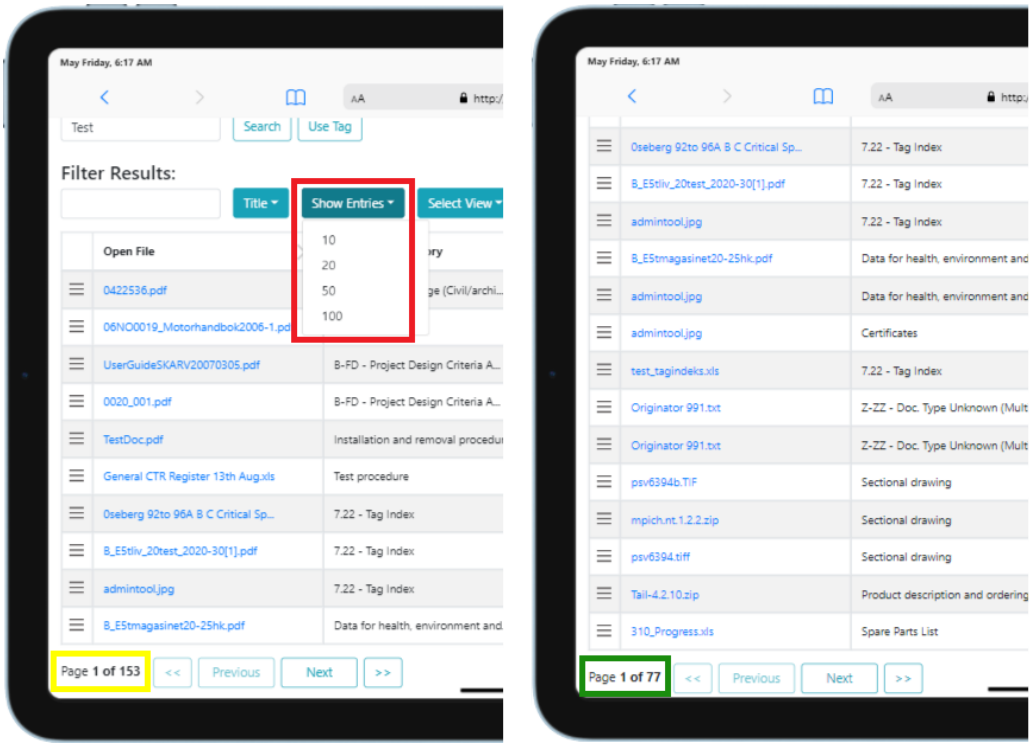
Figur 4.3.3 – Lokalt Filter

Markert i rødt, ser en det lokale filteret basert på søkeordet «General» i «Title» kolonnen. Knappen til høyre, markert både i det rødt og lilla, er hvor brukeren har mulighet til å spesifisere hvilken kolonne søket skal gjøres på. Standardsøket blir gjort på alle kolonner. Markert i grønt, kan en se at alle titler inneholder søkeordet «General» etter at søket har blitt gjennomført.

Endre antall rader

Standardoppsettet, viser kun 10 rader eller instanser per side. Brukeren har deretter mulighet for å bla gjennom resultatene med en meny under tabellen. Ettersom det kan være flere tusen rader, kan brukeren ha ønsker om å endre antall rader som blir vist på hver side. Dette kan bli gjort ved bruk av «Show Entries» knappen, som gir brukeren en dropdown meny med ulike alternativer.

Figur 4.3.4, viser hvordan antall rader og sider endres etter valg av «Show Entries» markert i rødt. I dette eksempelet, ble antall rader endret fra 10 til 20. Markert i gult, kan en se antall sider før endring, som her var 153. Etter endring (markert i grønt) er det kun 77 sider.

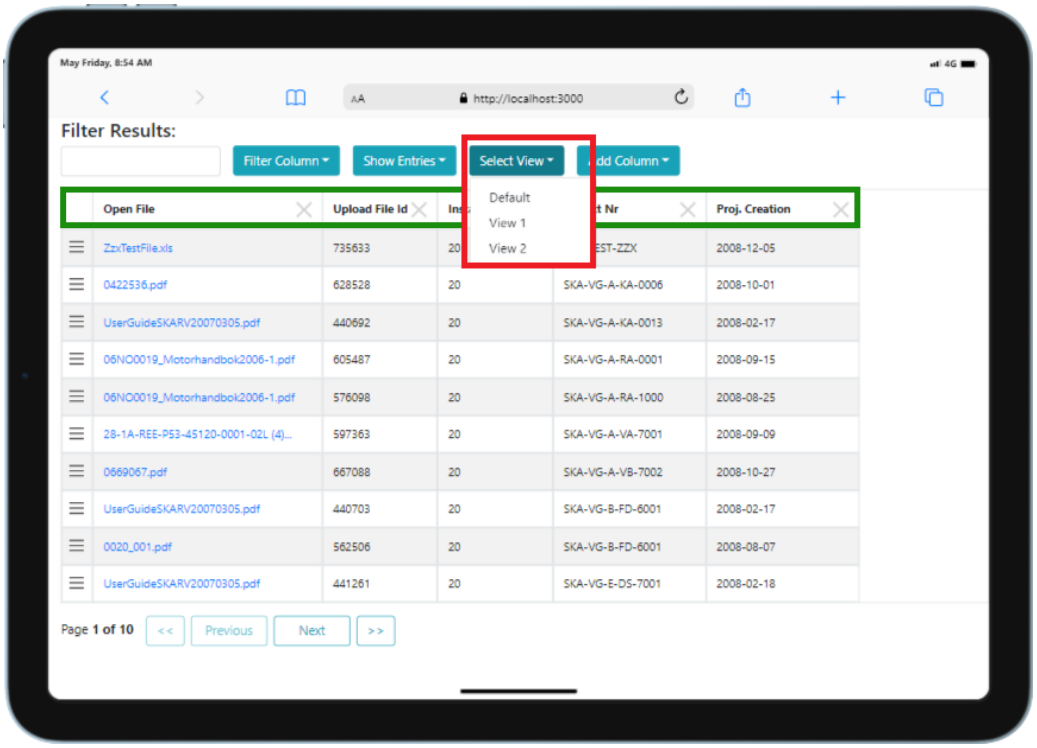


Figur 4.3.4 – Endre antall rader

Valg av kolonneoppsett

Ettersom applikasjonen er i utgangspunktet tenkt som en mobilapplikasjon, er kun noen få kolonner med i standardoppsettet. Det kan allikevel være et behov for å vise frem andre kolonner og en av mulighetene her, er da å velge mellom predefinerte kolonneoppsett som her er bruker betegnelsen «Views».

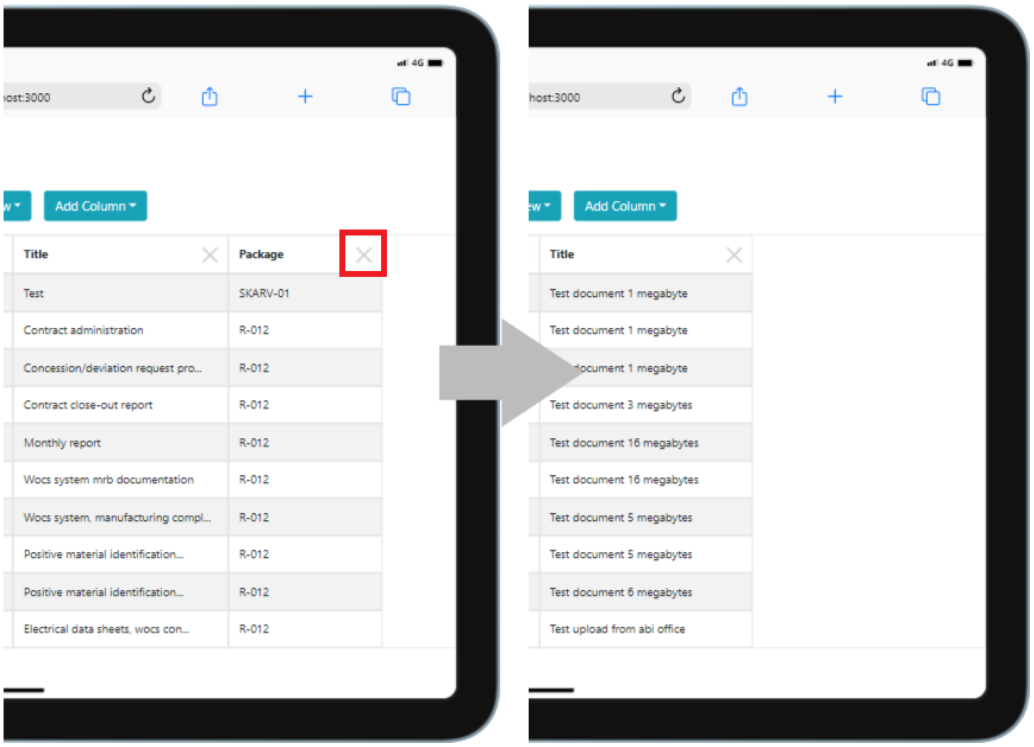
Figur 4.3.5 viser markert i rødt, en dropdown meny «Select View» hvor en kan velge mellom 3 predefinerte oppsett. I dette eksempelet, ble «View 1» valgt og kolonnene ble oppdatert som vist i grønt.



Figur 4.3.5 – Endre kolonneoppsett

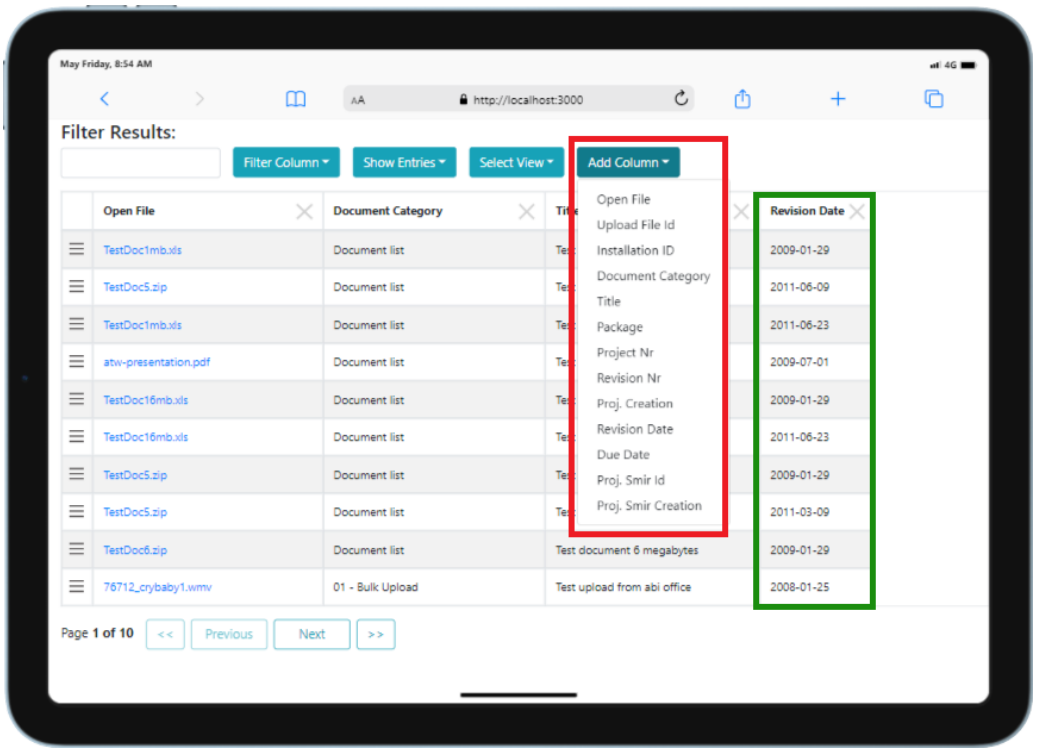
Egendefinert kolonne konfigurasjon

Ettersom de predefinerte kolonne oppsettene kanskje ikke er tilstrekkelig for alle brukere, er det muligheter for å definere individuelle kolonneoppsettet. For å fjerne kolonner, har brukeren mulighet til å trykke på «X» til høyre for hvert kolonnenavn. Dette er markert som rødt i figur 4.3.6.



Figur 4.3.6 – Fjerne kolonne

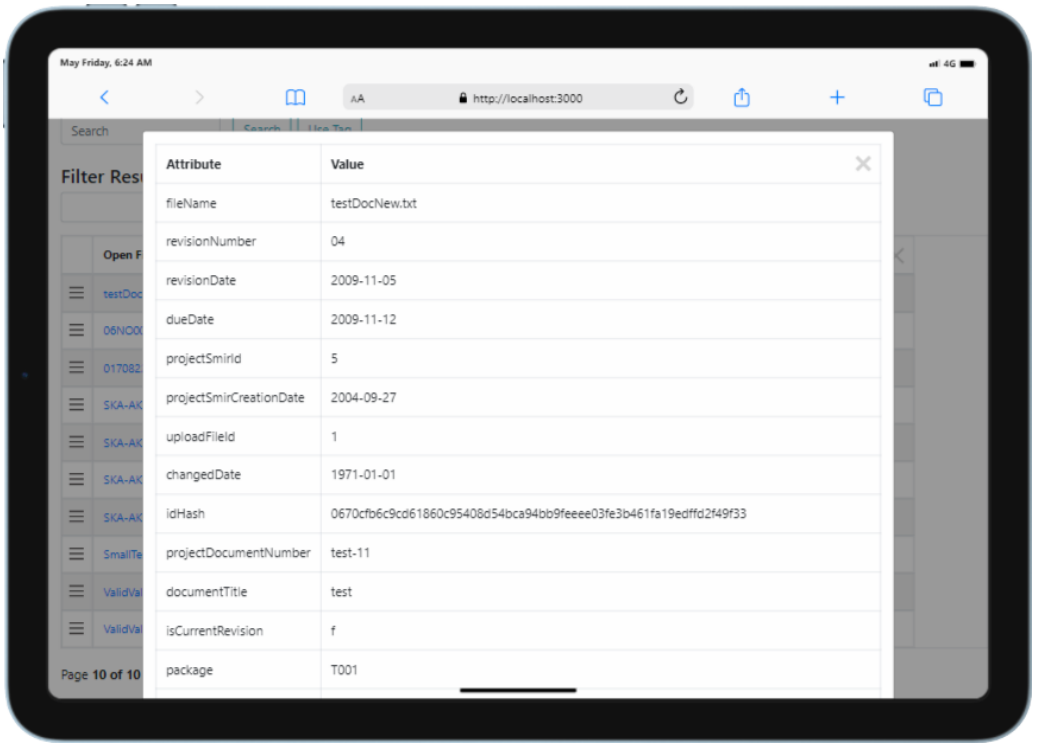
Når en har fjernet kolonner, kan det også være et behov for å legge til kolonner. Dette kan en gjøre ved bruk av «Add Column» dropdown menyen markert i rødt i figur 4.3.7. Brukeren får da mulighet til å velge mellom alle tilgjengelige kolonner og få den lagt til tabellen. I dette eksempelet, ble «Revision Date» lagt til.



Figur 4.3.7 – Legge til kolonne

Se alle attributter

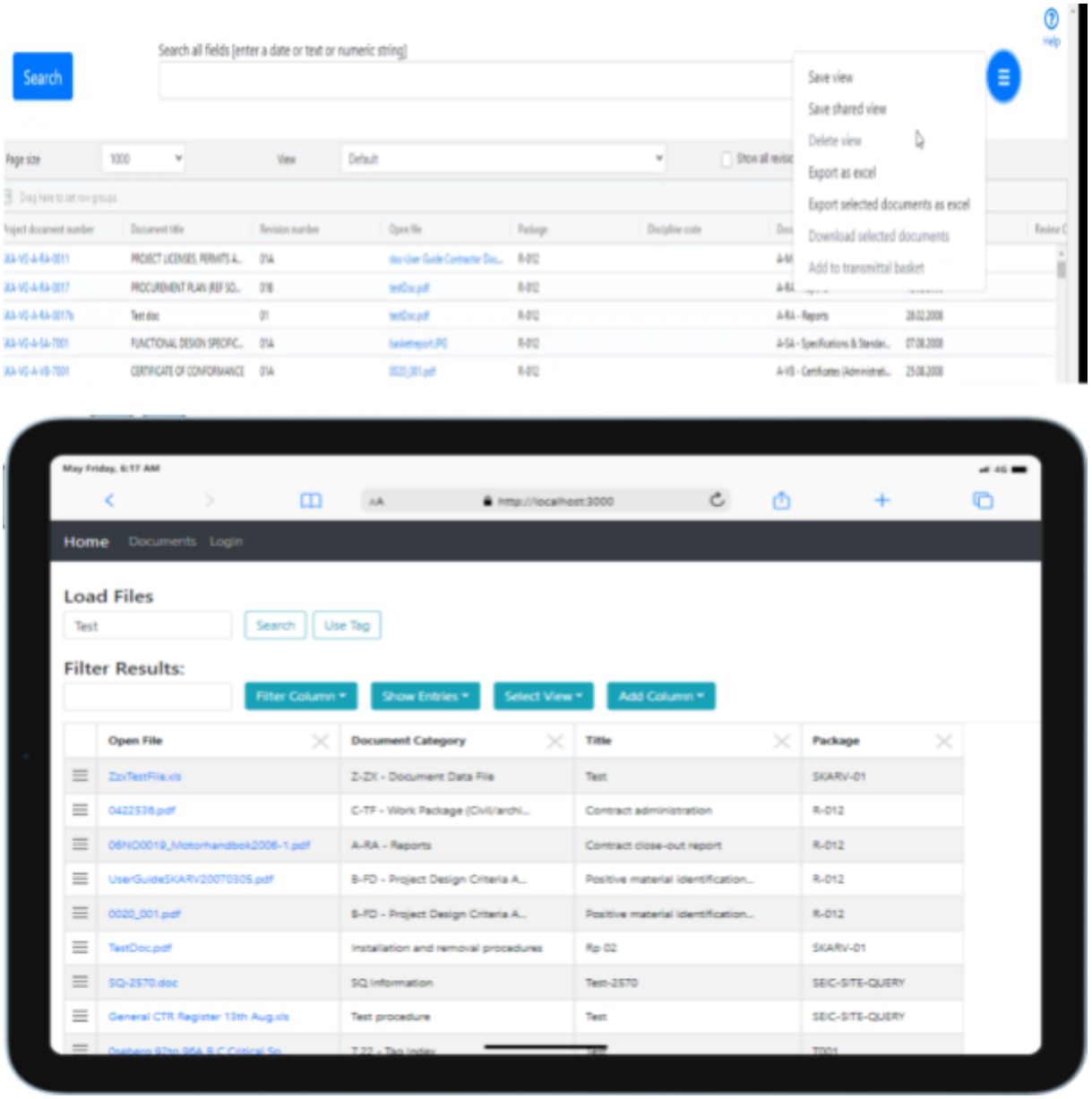
Tabellformatet er bra for å få en oversikt og filtrere data, men ettersom det kun kan være noen få kolonner om gangen, er det et behov for å kunne se mer fullstendig informasjon om instansene. Dette kan gjøres ved å klikke på symbolet helt til venstre i hver rad. Dette vil åpne et vindu med alle attributter og verdier for den instansen. Dette kan ses i Figur 4.3.8.



Figur 4.3.8 – Utdypende informasjon

9.4 ETC

9.4.1 Desktop applikasjon mot mobilapplikasjon



På det øverste bildet kan en se hvordan Sharecat sin eksisterende desktop applikasjon ser ut, mens på bildet under kan en se løsningen til gruppen. Desktop applikasjonen er lite egnet til mobile enheter og har mye informasjon pakket sammen på liten plass, tabellen en kan se på det øverste bildet viser bare noen få av kolonnene fra hele tabellen. Dersom en vil se flere kolonner må brukeren bla til siden for å se flere.

I løsningen til gruppen er dette problemet løst med å lage tabell radene større slik at de er lettere å lese på en mindre skjerm. Dersom en bruker vil se mer kolonner eller info om dokumentasjonen kan brukeren klikke på symbolet helt til venstre slik som vist i kapittel 4.3.