Systemdokumentasjon – IT-Drift Dashboard

1. Introduksjon

Denne systemdokumentasjon, skal forklare hvordan min applikasjon er

Tech stack: React (frontend), ChartJS, Axios, Bootstrap, Azure.

Jeg har valgt min tech stack basert på både min egen kompetanse, med tanke på skalerbarhet, og sikkerhet. Det er svært viktig å utvikle en løsning som både er nyttig, og relevant for Gabler, og gir verdi til bedriften.

2. **Brukerbehov og mål**
    a. **Brukerbehov:**

    Jeg har hovedsakelig jobbet med IT-Drift under læretiden, og jeg kom på at det hadde vært nyttig for meg å ha en slags oversikt over hvilke typer saker som vi oftest mottar. Et Dashboard er enkelt, og oversiktlig, samt intuitivt for brukeren for å raskt skaffe seg en oversikt over volumet av saker filtrert på applikasjoner. Et Dashboard gjør det også enkelt for ledelsen i Gabler å ta gode og overveide beslutninger basert på statistikk, ikke følelser. Brukerne blir de ansatte på IT-Drift, CTO i Gabler, og de andre i Ledergruppen. Jeg ser for meg at dashboardet kan bli brukt i anskaffelsesprosseser av nye applikasjoner, og vurderinger av nytteverdien en ny applikasjon har hatt, både fra tilbakemelding av brukerne, samt volumet av saker for applikasjonen.

    b. **Mål:**

    IT-Drift Dashboard er utviklet for å:

    - Gi en visuell og intuitiv oversikt over antall helpdeskhenvendelser per applikasjon.
    - Forenkle og effektivisere rapportering fra IT-Drift.
    - Underbygge anskaffelser og vurdering av nytteverdi basert på bruk, og saksmengde.
    - Tilby et verktøy som kan brukes i ledermøter, forbedringsprosjekter og analyse.

3. **Teknisk løsning:**

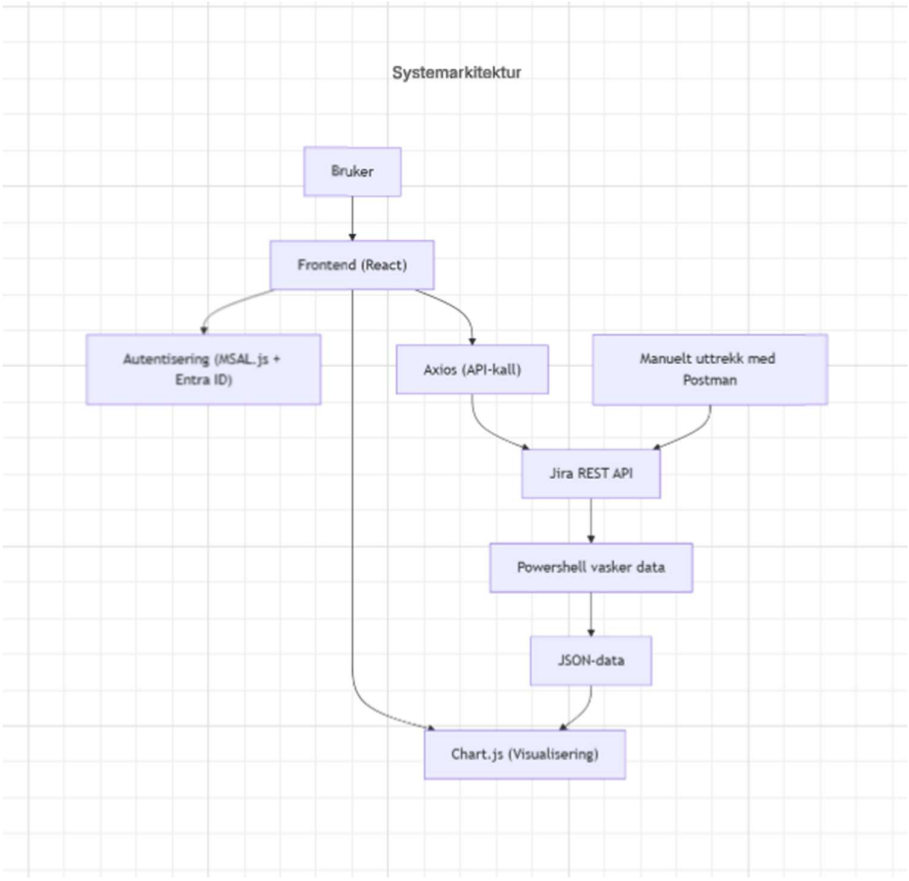
a. **Teknologistack**

Komponent	**Teknologi**
Frontend	React
Grafbibliotek	ChartJS
API-kall	Axios
UI-rammeverk	Bootstrap
Vask av API-data	Powershell
API	Jira REST API
Hosting	Azure Static Web App
Versjonskontroll	Github

b. **Systemarkitektur**

Løsningen min bruker React som frontend, ChartJS for datavisualisering, Axios for å hente data fra en JSON fil, som er hentet fra Jira Rest API med Postman, og vasket med Powershell, slik at personinformasjon og sensitive data er fjernet, i henhold til GDPR og DORA. Løsningen er hostet på Azure Static Web App, versjonskontroll håndteres med GitHub. Dette sikrer åpenhet og gjør det enkelt for andre utviklere å skalere lløsningen ytterligere. Brukere kan filtrere data etter nøkkelord og tidsintervaller. De mest relevante tidsintervallene for dette prosjektet, er siste dag, 7 dager, 30 dager. Systemet er utviklet med et fokus på sikkerhet, skalerbarhet og brukervennlighet.

Dette diagrammet viser hvordan komponentene i løsningen henger sammen:



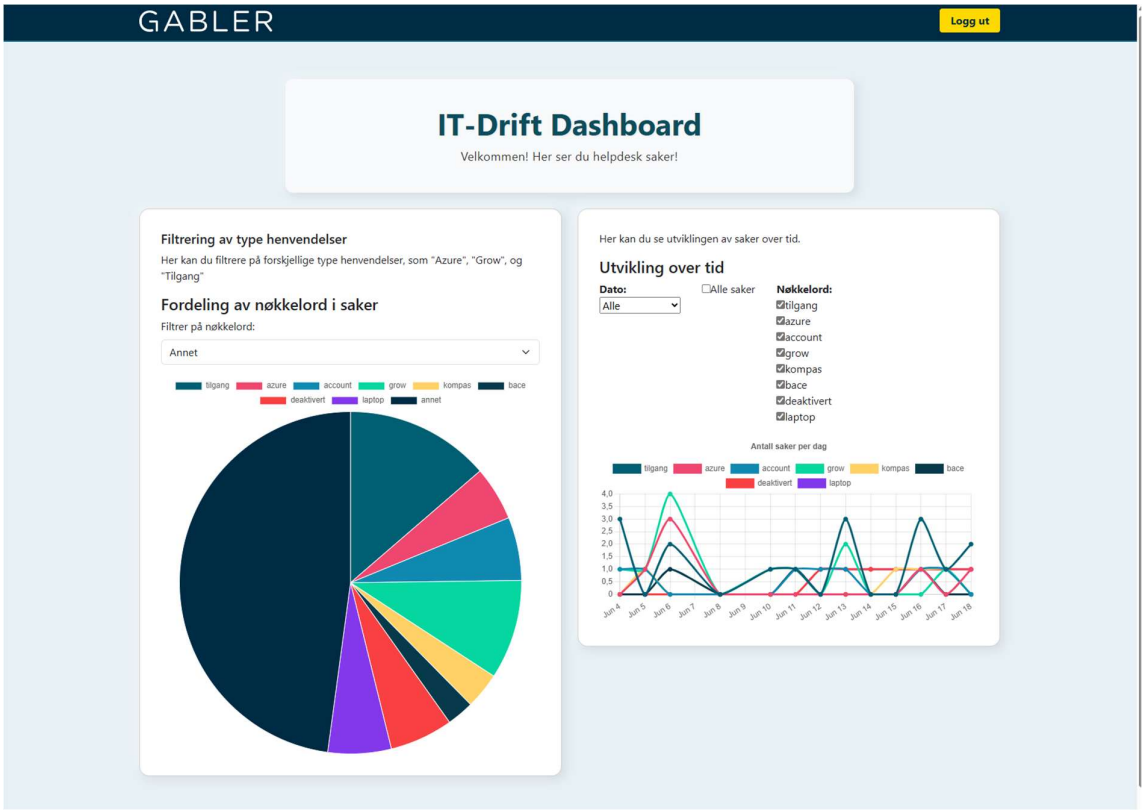
Løsningen er tilgangstyrt med Azure Group. For å få tilgang til applikasjonen, må du være medlem av entra gruppen «itd-helpdeskdashboard-user». Jeg har bare definert App.User i første iterasjon av løsningen, men i fremtidige releases vil det være et admin grensesnitt som CTO og IT-Drift Leder har tilgang til, hvor de kan se hvem som bruker dashboardet.

4. **Funksjonalitet & Brukergrensesnitt**

   Hovedfunksjonene til Dashboardet er å vise volum av helpdeskhenvendelser, filtrert på nøkkelord. Nøkkelordene er nå «Account, Grow, Bace, Kompas, Tilgang, Deaktivert og Laptop». Det er også en egen kategori for «Annet», hvor saker som ikke har disse nøkkelordene havner. I første iterasjon av løsningen, gjorde jeg et uttrekk av

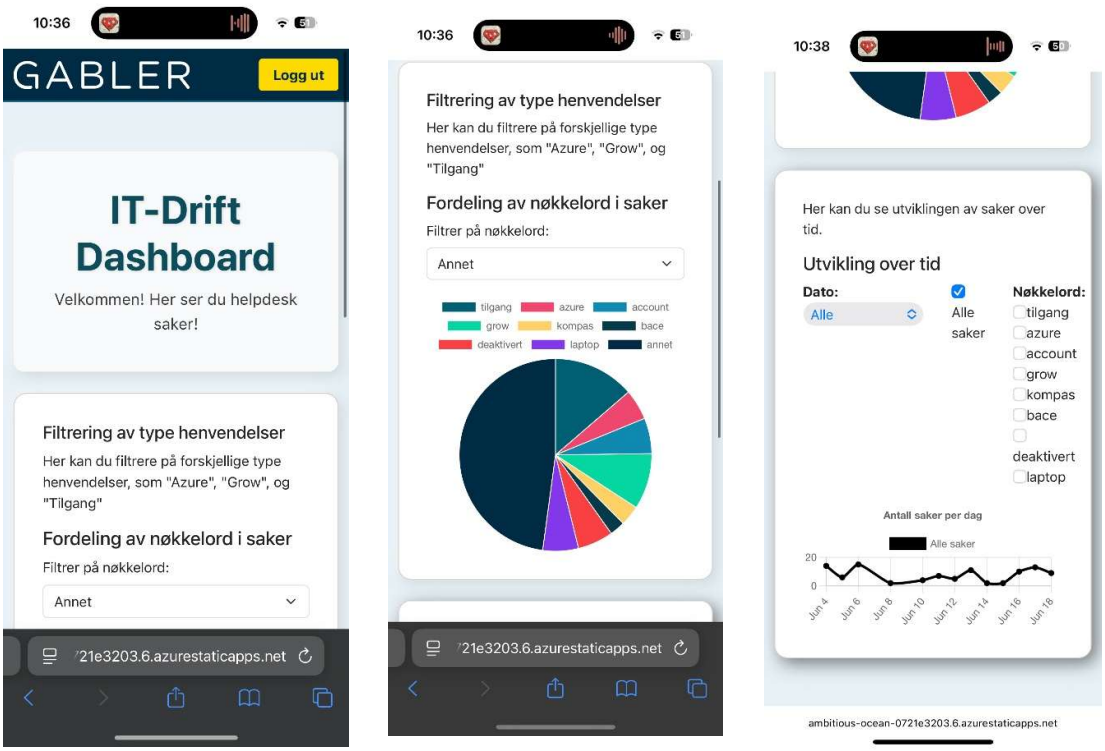
de siste 100 sakene den 18. juni 16:00. I en Prod-løsning ville jeg brukt Azure Functions til å koble til Jira REST API, hente ut data til en JSON fil, og vaske med Powershell for å etterfølge GDPR og DORA.

Det er en enkel og intuitiv visning av dataene hentet fra json filen, visualisert med ChartJS. Med en piechart som viser fordelingen av nøkkelord i saker, samt et linjediagram som viser volumet av saker over tid.



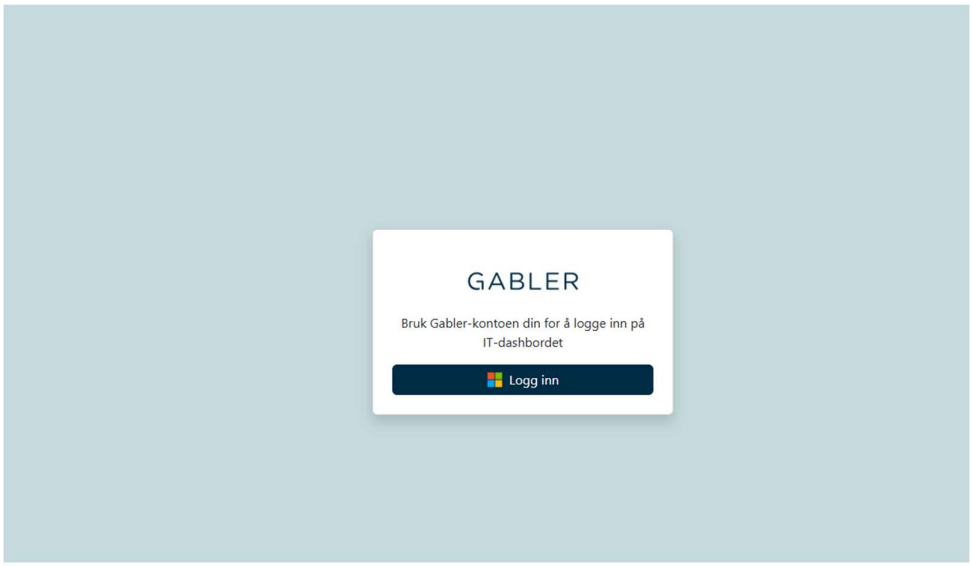
Det er en chart som mangler, grunnet begrenset med tid og nødvendig avgrensning, nemlig en bar chart som viser status på saker, «in progress», «done» og «not started». Bar charten vil ha et lignende filter, men med mulighet for å velge konkrete datoer, slik at du enkelt kan se hvor mange saker som hvilken status.

Det var også viktig for meg under utviklingsprosessen, at Dashboardet skulle være tilgjengelig på alle plattformer, derfor valgte jeg å benytte Bootstrap, som for meg er synonymt med responsivt design. Kombinert med hosting i Azure Static Web App, fører det til en løsning som er tilgjengelig på Telefon, Nettbrett, og PC.



Det er en enkel brukerflyt, den starter med en loginside, hvor du trykker på «logg inn». Jeg har brukt MSAL til å få Entra ID Authentication, og som nevnt tidligere, er det tilgangstyrt med en Entra Gruppe. Jeg har også satt det opp slik at brukeren din må være en del av Gabler sin Entra tenant for å få tilgang, for å forhindre at uvedkommende får tilgang. Dersom du er utenfor Gabler sitt nett, eller på en ukjent enhet, må du også verifisere med MFA for å få

tilgang. Dette er et krav som er helt essensielt, og er svært viktig. Slik ser Login siden ut:



Dersom du ikke er medlem av «itd-helpdeskDashboard-user», vil du få en slik feilmelding

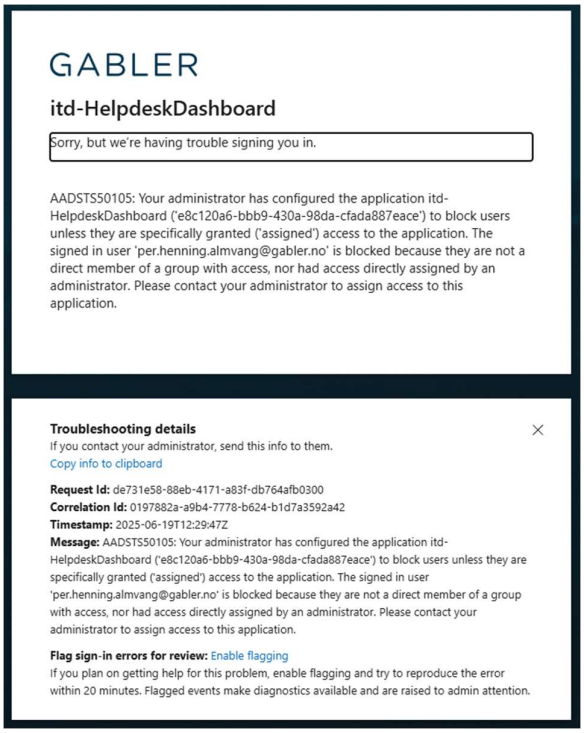
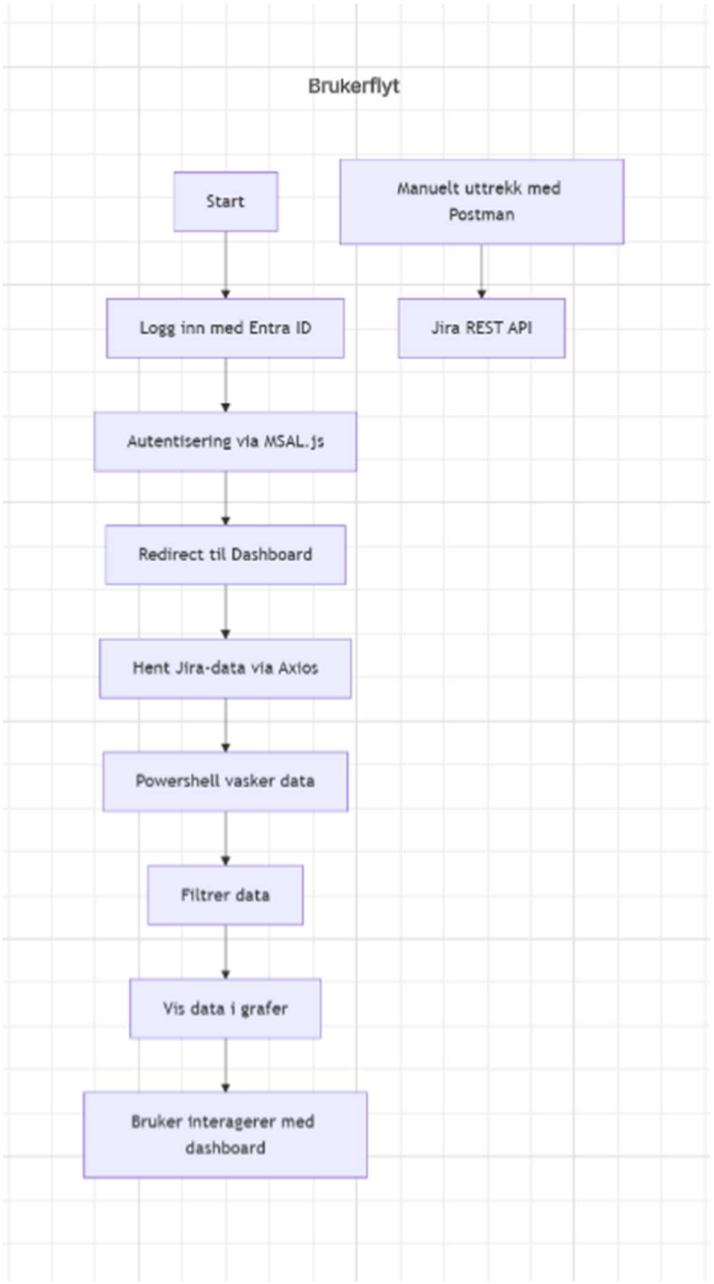
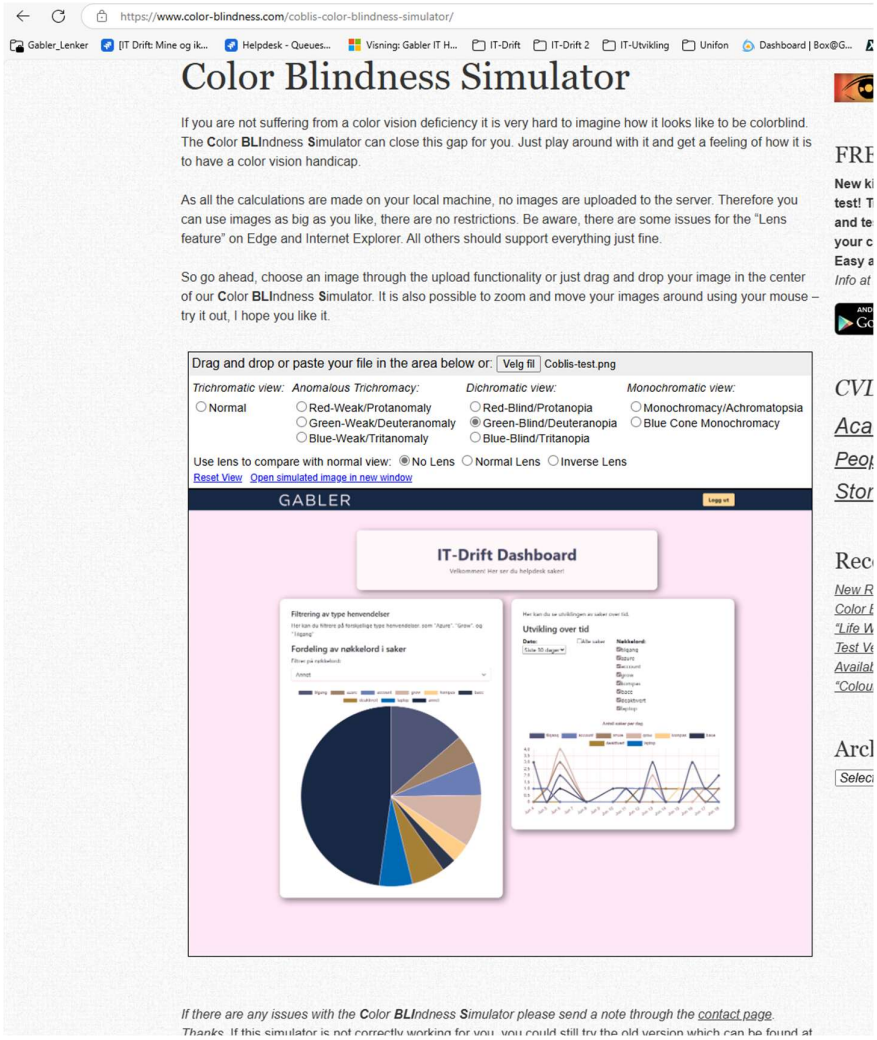


Diagram 3: Brukerflyt

Dette diagrammet viser hvordan en bruker navigerer fra innlogging til interaksjon med dashboardet.



Jeg har også testet at fargene mine er synlige selv for brukere med fargeblindhet. Dette gjorde jeg ved å bruke «Coblis – Color Blindness Simulator».



Jeg må være ærlig og si at det ikke er alle «Color blind filterne» hvor det er god nok kontrast mellom fargene. Det er et forbedringspotensiale, og svært viktig for Universell utforming.

5. **Sikkerhetsvurdering**

Sikkerhet er som nevnt en svært viktig del av hvert IT-Prosjekt. Jeg har derfor gjort et par konkrete tiltak. I min opprinnelige plan hadde jeg tenkt å simulere autentisering, men etter hvert oppdaget jeg at jeg hadde tid til å sette opp Entra ID Autentisering, som var min opprinnelige plan. Dette gjør at jeg fyller opp en rekke krav, blant annet forhindrer dette Uautorisert tilgang til data, fordi man ikke kommer seg inn i appen uten autentisering via Entra. Løsningen benytter også Entra sin innebygde tilgangslogg, dette gjør det mulig å se hvem som har logget seg inn på applikasjon, hvor, og når

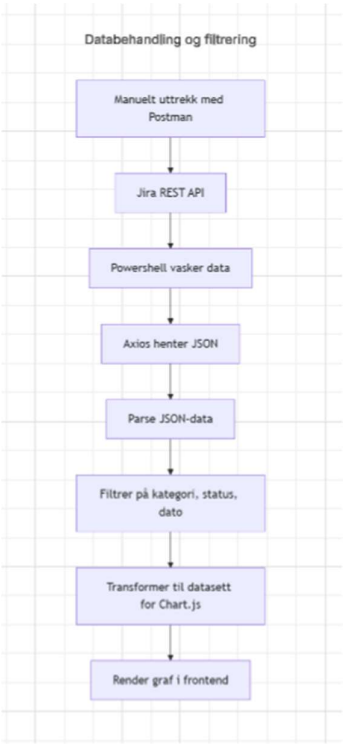
En Azure Static Web App har også tvunget HTTPS. Det er ingen sensitive nøkler, eller tokens som lagres i Frontend, disse er definert i Azure. Bibliotekene jeg har brukt, som Axios, ChartJS, og Bootstrap, er oppdatert og verifisert. Jeg benytter også GitHub, som gir meg og andre utviklere kontroll på endringer.

Persondata og etterlevelse (GDPR, DORA)

- GDPR (General Data Protection Regulation) gjelder for alle organisasjoner som samler inn, behandler eller lagrer personopplysninger om enkeltpersoner, innenfor EU og EØS. Personopplysning er definert som enhver info om identifiserbar person. Det mest relevante eksemplet i mitt tilfelle, er epost-adresse, navn på reporter og evt navn i subject i saken. F. eks: «Morten må ha en ny PC». Ingen skal kunne aksessere denne dataen i løsningen min, men hvordan har jeg løst dette? For dette prosjektet, måtte jeg gjøre et par avgrensninger, jeg rakk ikke å etablere en live connection med Jira REST API, så jeg har gjort uttrekk med Postman. Deretter har jeg et Powershell script som fjerner irrelevant data. I mitt tilfelle trengte jeg bare dato, status og subject. All annen info, slik som email, reporter, assignee, blir slettet og ikke tatt med. Da er vi GDPR Compliant.

Diagram 2: Databehandling og filtrering

Dette diagrammet viser hvordan Jira data hentes, parses, filtres og transformeres før det vises i dashboardet.



- **Hvordan har valg av tjenester blitt påvirket av GDPR og DORA?**

  Valget om å bruke **Entra ID** for autentisering var først og fremst drevet av to hensyn: For det første er Gabler en Azure-basert virksomhet, noe som gjør det naturlig å velge en løsning som er tett integrert med eksisterende infrastruktur. Det forenkler utvikling, vedlikehold og tilgangsstyring betydelig. For det andre gir det en bedre brukeropplevelse – ansatte kan logge inn med sin eksisterende Gabler-konto uten å forholde seg til nye brukernavn og passord.
  Selv om Microsoft som leverandør har godt dokumenterte rutiner for etterlevelse av både **GDPR** og **DORA**, er det viktig å understreke at **compliance ikke er noe man kan "arve" fra en plattform**. Ansvaret for å håndtere personopplysninger sikkert og i tråd med regelverket ligger fortsatt hos den som utvikler og drifter løsningen.
  Derfor har jeg gjort flere konkrete tiltak for å sikre etterlevelse:
  Minimering av persondata: Kun nødvendige felter (dato, status og subject) hentes fra Jira. Felter som e-post, reporter og assignee ekskluderes og slettes før videre bruk.
  Tilgangsstyring: Autentiseringen bruker Entra ID med rollebasert tilgang (RBAC), slik at kun godkjente brukere får tilgang til dashboardet.
  Loggføring og kontroll: Systemet er satt opp slik at brukeraksess kan etterspores uten at sensitive data eksponeres, dette gjøres i Entra, i selve Enterprise application, med sign in logs.
  Dokumentasjon og vurdering: Valget av tjenester og databehandlingen er dokumentert og vurdert med tanke på risiko og kontinuitet, som krevd i DORA.

OAuth 2.0 / OpenID Connect:

OAuth gir god sikkerhet og er en åpen standard for autorisasjon som er mye brukt i webapplikasjoner. Den er enkel å implementere, spesielt via tredjepartsleverandører som Google eller Auth0. Ulempen er at dette medfører avhengighet til eksterne aktører utenfor Gablers kontroll, noe som kan gjøre etterlevelse av GDPR og DORA mer utfordrende. I tillegg hadde det ført til en mindre sømløs opplevelse for sluttbrukerne, da de måtte logge inn med en annen identitet enn sin Gabler-konto.

Single Sign-On (SSO) med Okta:

Okta er en ledende plattform for identitets- og tilgangsstyring. Den tilbyr funksjonalitet tilsvarende Entra ID, inkludert SSO, RBAC og MFA. Likevel ble den vurdert som mindre hensiktsmessig i dette prosjektet, fordi Gabler allerede er tungt integrert med Microsoft Azure. Å introdusere Okta ville medført økt kompleksitet i brukeradministrasjon, behov for synkronisering av brukere, og potensielt høyere

kostnader. For Gabler ville dette også oppleves som unødvendig dobbeltarbeid for brukerne, som allerede benytter Entra ID i sine daglige systemer.

Konklusjon:

Entra ID ble valgt fordi det gir en sømløs og sikker autentisering for Gablers ansatte, krever minimal ekstra administrasjon, og støtter både sikkerhets- og etterlevelseskravene i GDPR og DORA.

6. **Risikovurdering:**

Dette kapittelet bygger videre på sikkerhetsvurdering, det er viktig å identifisere og vurdere risikoer knyttet til løsningen. Under følger en oversikt over de truslene jeg mener er mest relevante, hva konsekvensene er, og hvilke tiltak jeg har implementert for å redusere risikoen.

Trussel 1: Uautorisert tilgang til Dashboardet:

Konsekvensene av uautorisert tilgang til dashboardet, er eksponering av bedriftssensitive data, og innsikt i IT miljøet til Gabler. Dette kan føre til at ondsinnede aktører har bedre muligheter for å gjennomføre et angrep mot Gabler, da de ser hvor svakhetene våre er.

Trussel 2: Eksponering av personopplysninger

Dersom vi eksponerer personopplysninger i dashboardet, fører dette til brudd på GDPR, som igjen fører til mulig bøtelegging. Det fører også til tap av tillit mellom IT-Avdelingen, og brukerne. Tiltaket jeg har gjort, er å vaske dataen med Powershell før bruk, det gjør at kun nødvendige felter (dato, status og subject), beholdes. Dette er også de eneste feltene som er definert i koden.

Trussel 3: Feil i visualisering i grafene

Dette fører til feil beslutningsgrunnlag for ledelsen, og kan føre til dårlige økonomiske beslutninger som forhindrer verdiskapning for Gabler. Tiltaket som er utført er at filtreringen skjer i frontend.

Trussel 4: Bruk av utdaterte biblioteker

Potensielle sikkerhetshull er den største trusselen her, jeg har oppdatert alt til siste stabile versjon (se teknologistacken), og bruker GitHub for versjonskontroll.

Testing og Kvalitetssikring

For å sikre at løsningen fungerer som forventer, har jeg denne gang utført manuell funksjonstesting. Dersom jeg hadde hatt bedre tid, hadde jeg brukt Cypress for å automatisere dette. Under kommer en liste over manuelle tester jeg har gjennomført

Funksjonell testing:

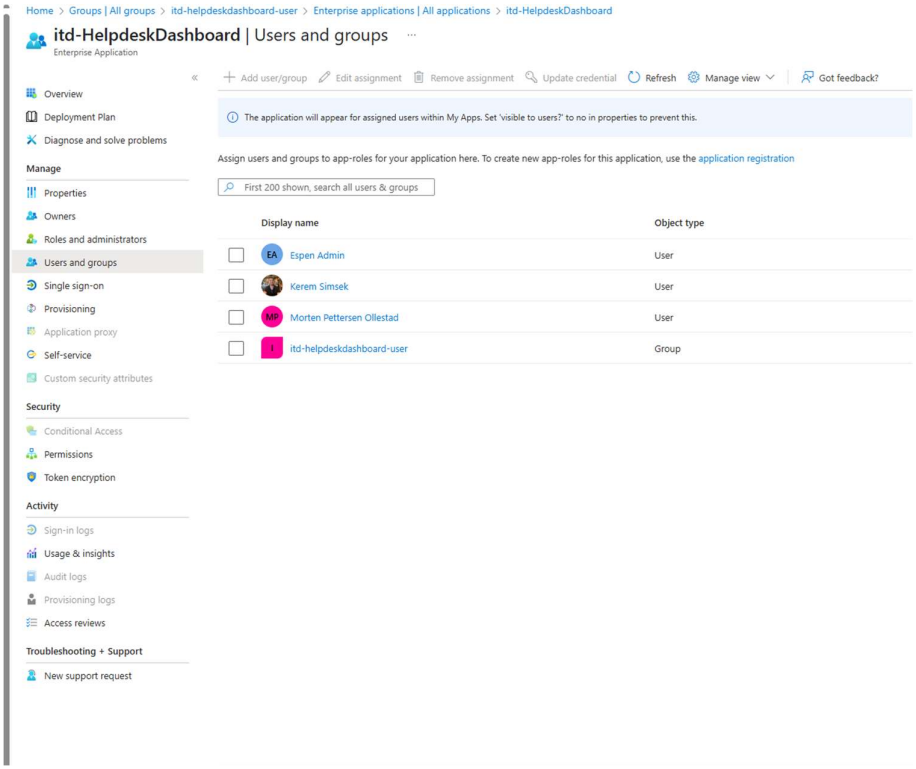
Filtrering på nøkkelord er testet i piechart og linjediagram. Filtrering på tidsintervall er testet på Linecharten, da en slik funksjonalitet ikke er lagt inn på piecharten, da jeg måtte avgrense omfanget av prosjektet litt.

Innlogging med Entra ID og MFA er testet med ulike brukere og verifisert.

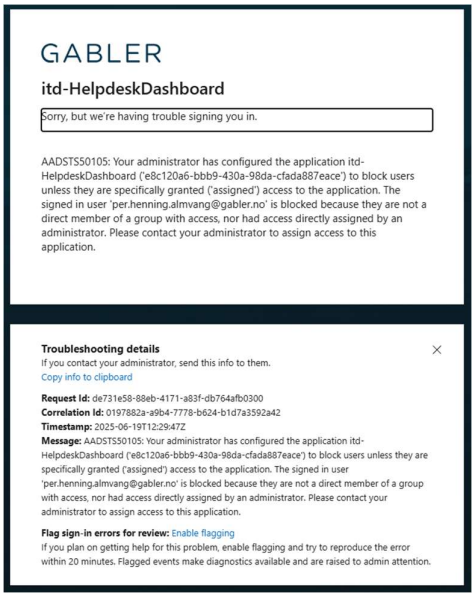
Responsivt design er testet på mobil, pc og nettbrett.

Sikkerhetstesting:

Verifisert at kun brukere i «itd-helpdeskdashboard-user» har tilgang til applikasjonen, slik det er definert i Entra Enterprise Application:



Dersom du ikke er medlem av gruppen, får du denne feilmeldingen:



Dette betyr at det fungerer slik det skal.

Siden appen er hostet via Azure Static Web App, blir HTTPS aktivert og tvunget. Azure Static Web App generer et gyldig SSL-sertfikat for det tilknyttede domenet, dette sørger for at alle forespørsler som kommer inn via http, blir omdirigert til HTTPS. Dette gir brukeren – og

applikasjonen en økt følelse av sikkerhet, siden man er beskyttet mot man-in-the-middle-angrep, uautorisert innsyn i datatrafikken, samt datalekkasjer.

Feilhåndtering:

Her er det ikke gjort så mye enda, men dersom Axios ikke klarer å hente data fra JSON.filen, vil ikke grafene vises, og det står «laster data».

Kvalitetssikring:

Jeg har under hele utviklingsprossessen fulgt best practices for React.
Dette inkluderer å strukturere koden i komponenter, og tenke på skalerbarhet. Det er noe jeg har hatt i fokus under utviklingen, at dette skal være enkelt og skalere, samt koble på det faktiske Jira REST AP slik at det blir oppdatert live.
README.MD er også oppdatert med avhengigheter, samt installasjonsintrukser.

Videreutvikling og forbedringspotensial

1. Med mer tid og ressurser ville jeg ha fullført integrasjonen med Jira REST API via Azure functions. Slik ville det fungert:

a. Azure Function (timer-trigger)
   En Serverløs funksjon i Azure, som kjører hver time. Jeg ville også implementert en «refresh» knapp i frontend-en, som trigger dette kallet.

b. Autentisering mot Jira API
   Funksjonen hadde brukt Basic Auth, med Jira API-token, som er lagret i Azure som en miljøvariabel, for å hente ut relevante saker

c. Datavask i funksjonen
   I dagens løsning vaskes dataen lokalt med PowerShell før den lastes inn i frontend. Dette fungerer godt for et avgrenset prosjekt, men i en produksjonsløsning ville det vært mer hensiktsmessig å flytte denne logikken til en Azure Function.

   Azure Function-en kunne da, etter å ha hentet data fra Jira REST API, kjørt en enkel filtreringslogikk direkte i koden (f.eks. i JavaScript eller C#). Denne funksjonen ville fjernet alle felter som inneholder personopplysninger, som reporter, assignee, email, eventuelle

personopplysninger i summary eller description. Kun nødvendige felter som created, status og summary ville blitt beholdt, og returnert som JSON til frontend.

d. Lagring av data
   Den vaskede JSON-dataen kunne blitt lagret i Azure Blob Storage som en JSON fil, Axios ville da hentet data direkte fra Azure Blob Storage, i stedet for en statisk fil lokalt.

2. Jeg ville også implementert en Bar chart for å enkelt vise status på saker (Done, In Progress, Not Started). Dette ville gjort det enkelt å til enhver tid se hvor mange saker som har hvilken status, for IT-Drift Leder.

3. Lagt til et admin-grensesnitt for å se bruksmønstere, samt tilgangslogg for applikasjonen.

4. Lagt til «Søk etter nøkkelord» funksjonalitet, så man ikke er begrenset av de predefinerte nøkkelordene.

5. Utvide løsningen til andre helpdesklignende køer, for Gabler sin del er det mest aktuelle GROW, og henvendelser til Pensjonskassene Gabler administrerer.

Dekning av kompetansemål

I dette prosjektet har jeg bevisst jobbet for å dekke alle fem kjerneelementene i læreplanen for IT-utviklerfaget (IUV03-01), samt samtlige kompetansemål. Under følger en oversikt over hvordan disse er ivaretatt i løsningen:

Kjerneelementer

1. **Etikk, lovverk og yrkesutøvelse**
   - Jeg har etterlevd kravene i **GDPR** og **DORA** ved å vaske data for personopplysninger, og dokumentere hvordan løsningen håndterer sikkerhet og personvern.
   - Jeg har vurdert ulike autentiseringsløsninger (Entra ID, OAuth, Okta) og valgt den som best ivaretar sikkerhet og etterlevelse.
   - Jeg har reflektert over hvordan løsningen påvirker brukerne og bedriften, og hvordan den kan bidra til bedre beslutninger og mer effektiv drift.
2. **Kodeferdigheter og metode**
   - Jeg har utviklet løsningen i **React**, med bruk av **ChartJS**, **Axios**, og **Bootstrap**.
   - Jeg har strukturert koden i komponenter, brukt versjonskontroll (GitHub), og fulgt beste praksis for frontend-utvikling.
   - Jeg har brukt **iterativ utviklingsmetodikk** og dokumentert valg og endringer underveis.

3. **Sikkerhet og personvern**
    - Løsningen benytter **Entra ID** med MFA og rollebasert tilgangsstyring.
    - All data er vasket for personopplysninger, og det er implementert tiltak for å hindre uautorisert tilgang.
    - Jeg har dokumentert risikoer og tiltak i en egen sikkerhetsvurdering.
4. **Infrastruktur og arkitektur**
    - Løsningen er hostet i **Azure Static Web App**, med tanke på skalerbarhet og sikkerhet.
    - Jeg har vurdert og forsøkt integrasjon med **Azure Functions** og **Jira REST API**, og dokumentert utfordringer og forbedringspotensial.
    - Jeg har brukt miljøvariabler og vurdert hvordan løsningen kan driftes og videreutvikles i en profesjonell setting.
5. **Design, interaksjon og brukerdialog**
    - Jeg har brukt **Bootstrap** for å sikre responsivt og universelt utformet design.
    - Jeg har testet løsningen for fargeblindhet og gjort forbedringer for tilgjengelighet.
    - Brukergrensesnittet er intuitivt, med tydelig navigasjon, filtrering og visuell fremstilling av data.

Her er en oversikt over hvordan jeg har dekt hvert kompetansemål slik det står i læreplanen:

Kompetansemål	**Hvordan det er dekket**
Utforske og bruke utviklingsverktøy og programmeringsspråk	Brukt React, ChartJS, Axios, Bootstrap, GitHub
Bruke utviklingsmetoder og strategier	Iterativ utvikling, planlegging, testing og dokumentasjon
Bruke versjonskontroll og samarbeidsverktøy	GitHub med README, commit-historikk og struktur
Utforske og bruke ulike systemarkitekturer	Brukt klient-server-arkitektur, vurdert Azure Functions
Utforske og bruke ulike databaser og datakilder	Brukt JSON-data fra Jira REST API, vurdert live-integrasjon

Kompetansemål	Hvordan det er dekket
Bruke API-er og utvikle egne API-er	Brukt Jira REST API, forsøkt utvikling av Azure Function
Utforske og bruke ulike autentiserings- og autorisasjonsløsninger	Implementert Entra ID med MFA og RBAC
Utforske og bruke ulike sikkerhetsmekanismer	HTTPS, miljøvariabler, tilgangsstyring, datavask
Utforske og bruke prinsipper for universell utforming	Brukt Bootstrap, testet med fargeblindhetssimulator
Utforske og bruke prinsipper for bærekraftig utvikling	Løsningen bidrar til effektivisering og bedre ressursbruk
Utforske og bruke prinsipper for personvern og datasikkerhet	GDPR og DORA etterleves, dokumentert i sikkerhetsvurdering
Utforske og bruke prinsipper for teknisk gjeld og ressursbruk	Bevisst avgrensning og prioritering for å unngå teknisk gjeld
Utforske og bruke prinsipper for dokumentasjon og kvalitetssikring	Systemdokumentasjon, brukerveiledning, sikkerhetsvurdering
Utforske og bruke prinsipper for feilhåndtering og logging	Axios-feilhåndtering, vurdert logging i Entra
Utforske og bruke prinsipper for test og validering	Manuell testing, vurdert Cypress for automatisering
Utforske og bruke prinsipper for samhandling og samarbeid	Løsningen støtter samarbeid og beslutningsstøtte i Gabler
Utforske og bruke prinsipper for etisk refleksjon	Reflektert over personvern, sikkerhet og samfunnsansvar

Kompetansemål	Hvordan det er dekket
Utforske og bruke prinsipper for demokrati og medborgerskap	Løsningen fremmer åpenhet og innsikt i IT-drift
Utforske og bruke prinsipper for livslang læring og utvikling	Reflektert over læring, forbedringer og videreutvikling

Kilder

React – https://react.dev/

Chart.js – https://www.chartjs.org/

Axios – https://axios-http.com/

Bootstrap – https://getbootstrap.com/

Microsoft Entra ID (tidligere Azure AD) – https://learn.microsoft.com/en-us/entra/

MSAL.js – https://learn.microsoft.com/en-us/azure/active-directory/develop/msal-overview

Jira REST API – https://developer.atlassian.com/cloud/jira/platform/rest/v3/intro/

Azure Static Web Apps – https://learn.microsoft.com/en-us/azure/static-web-apps/

GDPR – https://www.datatilsynet.no/rettigheter-og-plikter/virksomhetenes-plikter/personvernforordningen-gdpr/

DORA – https://www.finanstilsynet.no/tema/ikt/dora/

Coblis – Color Blindness Simulator – https://www.color-blindness.com/coblis-color-blindness-simulator/

Læreplan for IT-utviklerfaget Vg3 – https://www.udir.no/lk20/iuv03-01