KOLLEKTIVET

gjør ditt kollektivliv enklere

Victoria Skjæret Strand

01.12.2025

Innholdsfortegnelse

1 Introduksjon

1.1 Konseptet

Se for deg at du nettopp har flyttet inn i et kollektiv der du ikke kjenner noen. Det er rotete overalt og det er ikke kjøpt inn verken oppvasksåpe eller tørkepapir. Senere på kvelden hører du at det er det fest i kollektivet, men du har ikke møtt på noen av de andre som bor der enda og har ikke fått en invitasjon. Du tør ikke slenge deg på og bestemmer deg for å dra i butikken i stedet og handle såpe og tørkepapir til kollektivet. Du har nå lagt ut for felleskostnader, men du har ingen måte å få de andre til å dele utgiften med deg. Hele dette scenarioet kunne sett helt annerledes ut dersom man hadde en plattform som KOLLEKTIVET.

De fleste som bor eller har bodd i kollektiv har trolig opplevd utfordringer knyttet til både kommunikasjon, renhold, innkjøp og mer. Hensikten med webapplikasjonen «KOLLEKTIVET» er å gjøre kollektivlivet enklere ved å samle alt dette på ett sted.

Dette er en webapplikasjon for alle typer kollektiv, og alle typer beboere. Om det så er et lite vennekollektiv der alle kjenner hverandre fra før, eller om det er et kjempestort kollektiv der ingen kjenner hverandre på forhånd.

1.2 Bruk av kunstig intelligens

Det er blitt brukt kunstig intelligens gjennom hele prosessen på lik linje som en ville brukt en foreleser eller en medstudent, for effektiviseringen av arbeidet. Det har gjennom hele vært bevissthet over å bruke det rett; som en støtte, ikke en fasit. Det har blitt brukt særlig til feilsøking under koding, idémyldring og systematisering i startfasen og som en ekstra ressurs for å «flette sammen» kodersnutter fra veiledninger på nett i de tilfeller der det har vært vanskelig å gjøre endringer spesifikt knyttet til mitt prosjekt kun ved bruk av veiledninger. Det er også blitt brukt som guide i forsøk på å hoste hele webapplikasjonen online med både frontend og backend, men dette var dessverre mislykket. Valgt KI-modell har vært ChatGPT av OpenAI (OpenAI, 2025).

1.3 Lenker

Prosjekter som bruker Java som programmeringsspråk og Spring Boot rammeverk er mer komplisert å hoste sammenlignet med eksempelvis PHP. Lenken til webapplikasjon vil derfor kun vise frontend-funksjonalitet per nå. Backend-delen er dokumentert i et eget GitHub-repository og demonstrert i YouTube-videoen lenket til nedenfor.

Webapplikasjonen: https://tittentei.eu/

GitHub: https://github.com/victoria-strand-dev/kollektivet_web_application_java

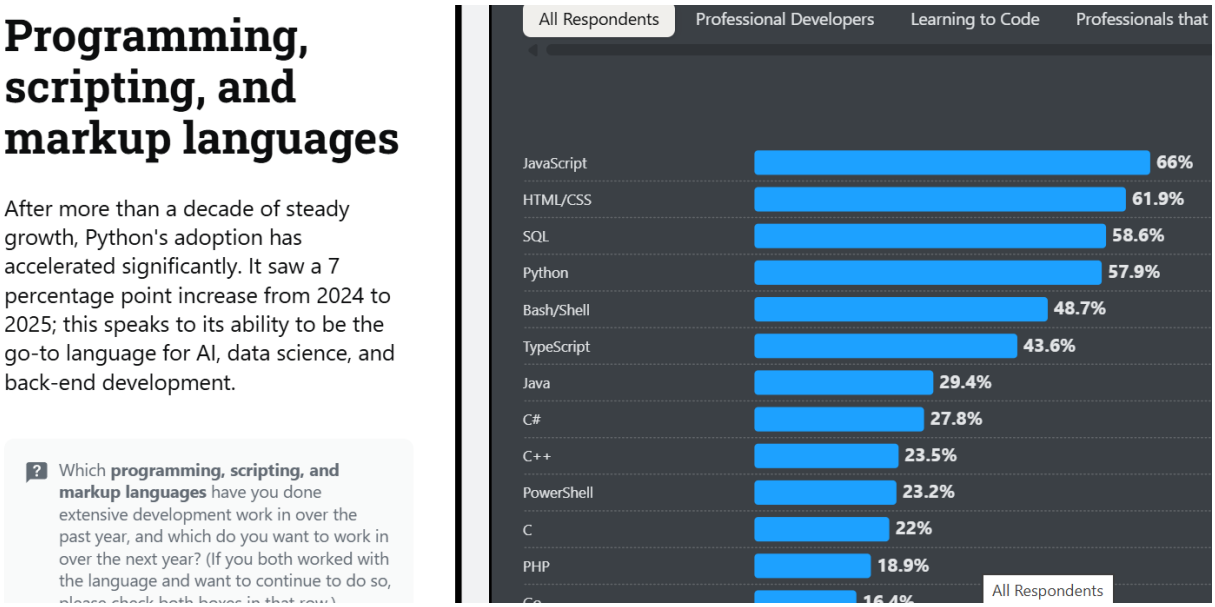
YouTube: https://www.youtube.com/watch?v=Q238C6y744g

2 Prosess

Startfasen i prosjektet bestod av idémyldring, innhenting av informasjon, sortering og endelige valg for de ulike delene i prosjektet. Det første og viktigste valget for oppstart var å finne ut *hva som skal lages*. Valget falt til slutt på en webapplikasjon for kollektiv.

Deretter måtte det bestemmes hvilke teknologier og verktøy som skulle brukes. Det ble først bestemt å bruke PHP og JavaScript i tillegg til HTML og CSS uten noe rammeverk, og Visual Studio Code som editor. Dette er enkel måte å kode webapplikasjoner på som funker og er veldig nybegynnervennlig.

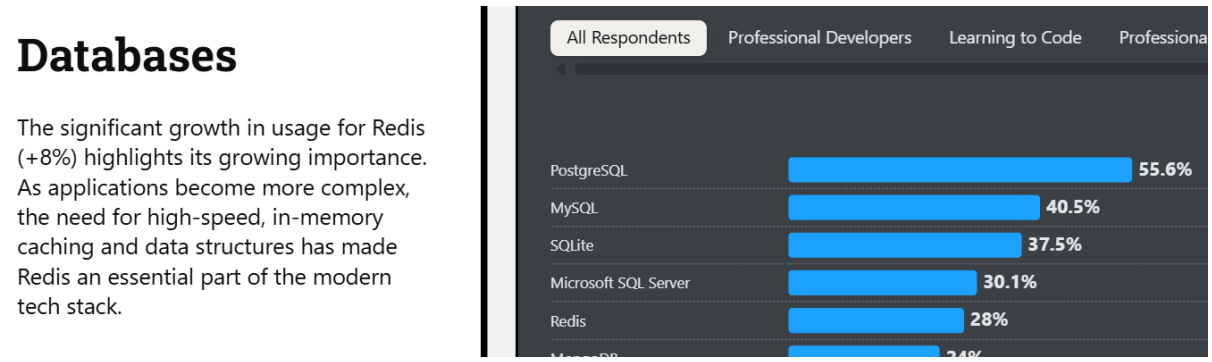
Prosjektet ble først utviklet basert på dette, men omtrent halvveis ut i prosjektet ble det bestemt å endre teknologier og verktøy. Årsaken til endringen var todelt. Hovedårsaken var dårlig struktur i prosjektet. Det var et ønske om å starte på nytt for å få en bedre struktur fra start. Da det uansett var aktuelt å starte på nytt ble det også interessant å titte på andre alternativer til teknologier og verktøyer.



Figur 1 – Skjermbilde fra Stack Overflow sin utviklerundersøkelse for 2025 over de mest populære programmerings-, scripting- og markupspråkene

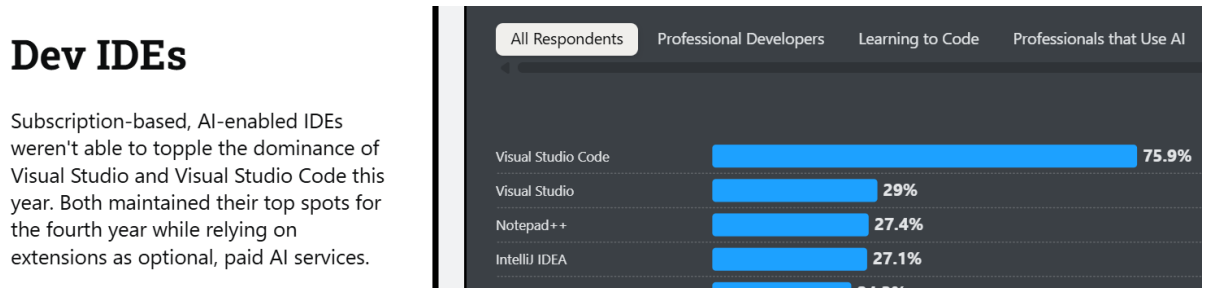
Figur 1 viser et skjermbilde fra Stack Overflow sin utviklerundersøkelse for 2025 (Stack Exchange Inc., 2025). Av undersøkelsen kommer det frem at PHP er mindre populært for webutvikling enn en rekke andre språk. Utenom JavaScript og SQL er Python øverst på denne listen, men basert på erfaring og ferdigheter fra tidligere ble det sammenlagt beste valget for dette prosjektet Java. Selv om Java ikke er fult så populært for webutvikling som Python, er det betydelig mer populært enn PHP. Det kan derfor være god erfaring å ha.

De videre valgene av teknologier og verktøy ble også tatt basert på statistikk, personlig erfaring og helhetsinntrykk fra søking på nett.



Figur 2 - Skjermbilde fra Stack Overflow sin utviklerundersøkelse for 2025 over de mest populære systemene for å administrere databaser

På figur 2 kan man se at PostgresSQL er det mest populære administrasjonssystemet for databaser. MySQL er listet som nummer to. Grunnet mer erfaring med MySQL, og MySQL Workbench, ble dette valget for databasen.



Figur 3 - Skjermbilde fra Stack Overflow sin utviklerundersøkelse for 2025 over de mest populære editorene og IDEene

Prosjektet ble først kodet i editoren Visual Studio Code. Av figur 3 kan man se at dette er langt mer populært enn alle andre editorer og IDEer (Integrated Development Environment). Det er fult mulig å bruke denne for webutvikling med Java også, men mange av veiledningene på nett for å sette opp et webutviklingsprosjekt med Java og Spring Boot bruker IntelliJ IDEA. Det virket derfor fornuftig og interessant å bruke det samme.

Da programmeringsspråket for den nye versjonen av prosjektet var bestemt, ble resten av teknologiene og verktøyene valgt naturlig gjennom

Teknologiene og verktøyene for den nye versjonen av prosjektet ble ikke aktivt valgt, men ble en naturlig del av prosjektet da det ble fulgt en guide laget av

Selv med ferdigheter innenfor Java fra tidligere, var det likevel nødvendig å finne informasjon for hvordan man bruker det i webutvikling. Gjennom søk på internett ble det tydelig at det er flere måter å sette opp et slikt prosjekt. Noe som gikk igjen var rammeverket Spring og Spring Boot.

Basert på grad av egen erfaring, statistikk hentet fra Stack Overflow sin utviklerundersøkelse for 2025 og innhenting av informasjon falt valget på programmeringsspråket Java, rammeverket Spring Boot, IDEen IntelliJ IDEA.

All CSS og ren HTML-kode kunne gjenbrukes, slik at det ikke var nødvendig å bygge opp alt helt fra bunnen av.

3 Java og Spring Boot

For å lage applikasjonen med Java har det vært gjort utallige søk på nett og blitt funnet ulike veiledninger. Hovedkildene til veiledning og kunnskap om hvordan ulike deler fungerer og henger sammen har vært Spring sine egne sider (VMware Tanzu, 2025) og nettstedet GeeksforGeeks (GeeksforGeeks, 2025).

4 System og kravspesifikasjon

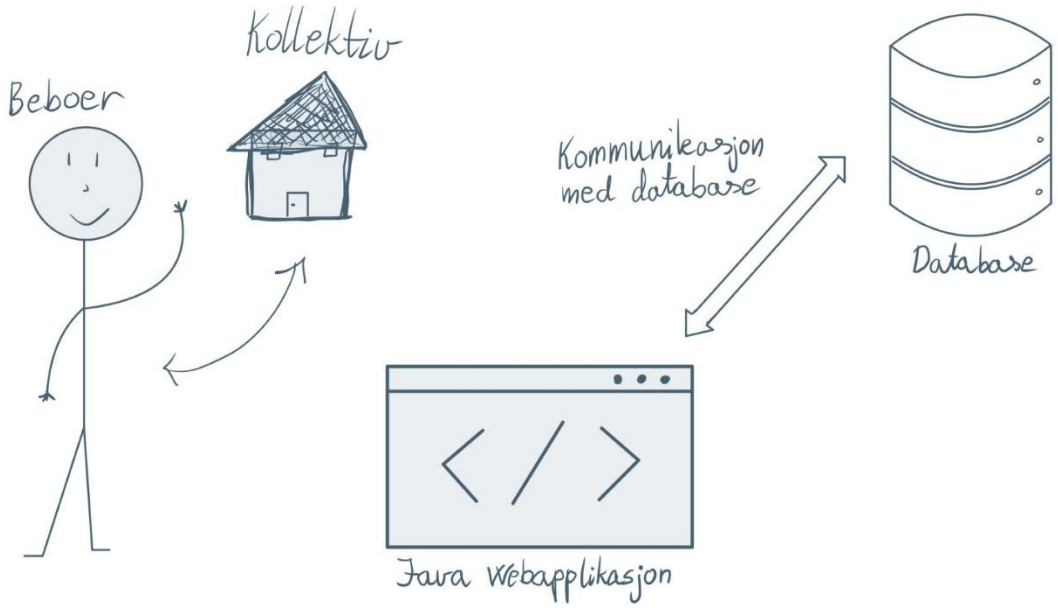
4.1 Overordnet beskrivelse av systemet

Jeg ønsker å lage en webapplikasjon som kan samle alt et kollektiv trenger på ett og samme sted.

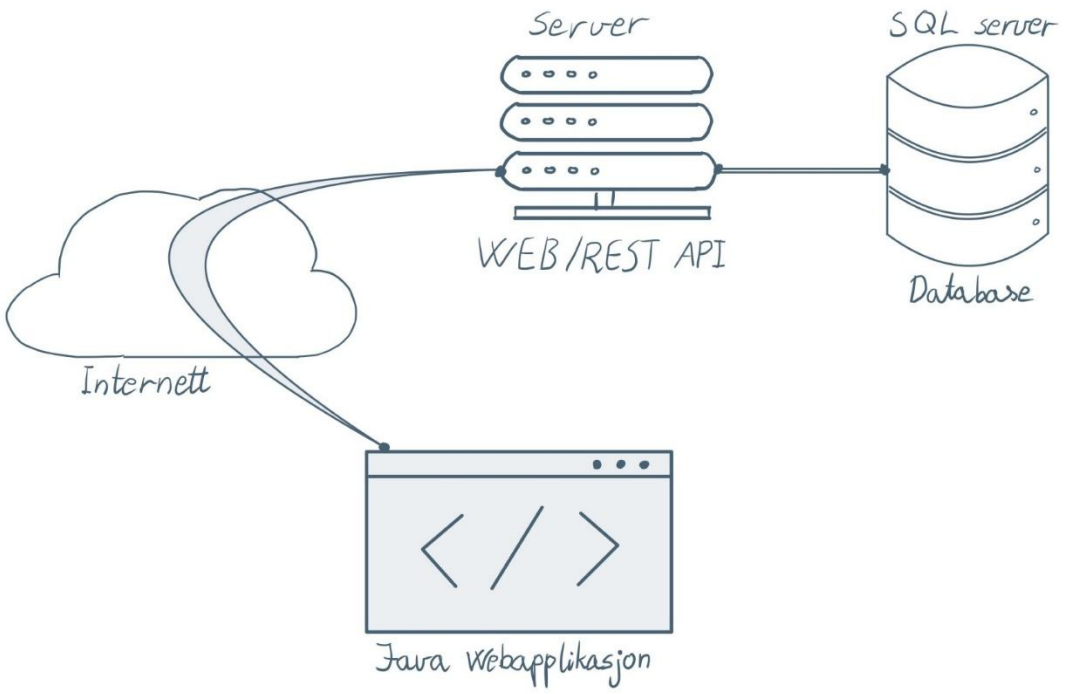
Handleliste, enkel informasjon om hvem som faktisk bor der (gjelder spesielt i store kollektiver), hvem som skylder hverandre hva (økonomi), kommende hendelser dersom det er en fest, en tacokveld eller et felles kjøkkenmøte, vaskeliste for å holde styr over hvem som har ansvar for hva, husregler lett tilgjengelig, en chat hvor man kan spørre om ting eller dele ting på ett sted fremfor å eksempelvis måtte ha Snapchat, Whatsapp eller Messenger lastet ned hos alle og måtte inviteres i en gruppe.

Jeg har et håp om at å samle alt på ett sted, samt å gjøre det mulig å delta i et kollektiv/få tilgang til informasjon/chater uten å måtte bli fysisk invitert kan gjøre kollektivlivet mer sømløst.

Figur 4 og 5 viser systemskissen og systemarkitekturen for webapplikasjonen.



Figur 4 - Systemskisse

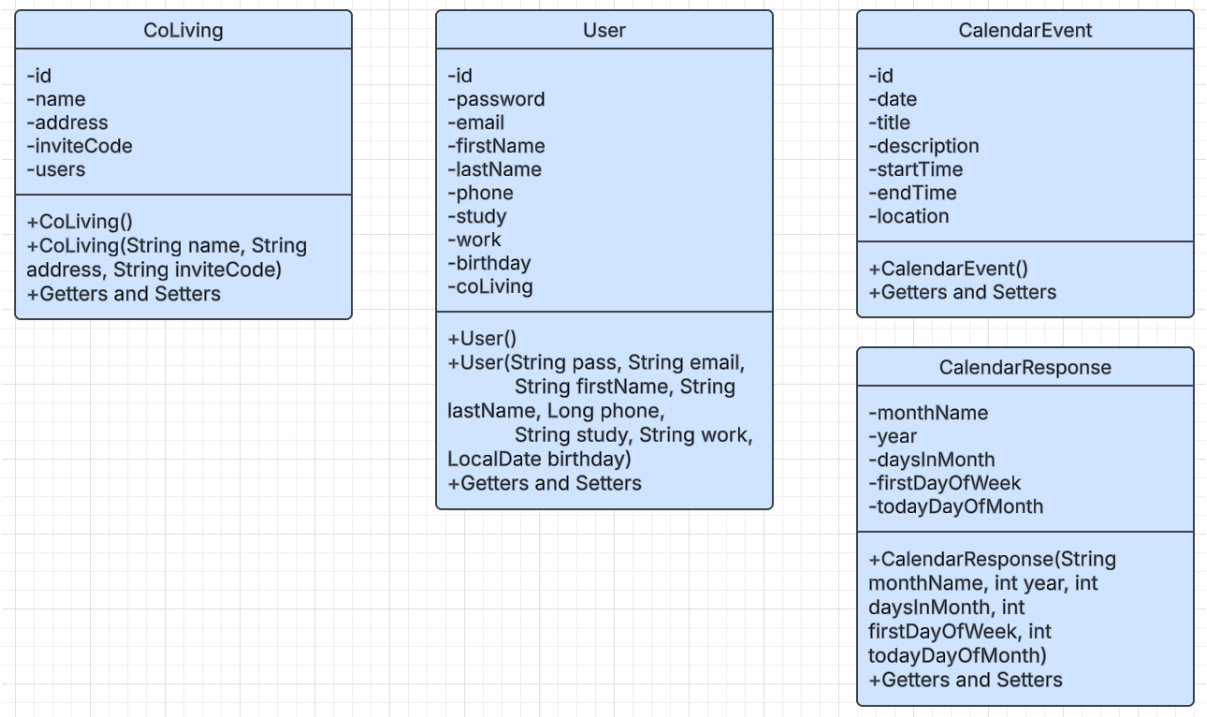


Figur 5 - Systemarkitektur

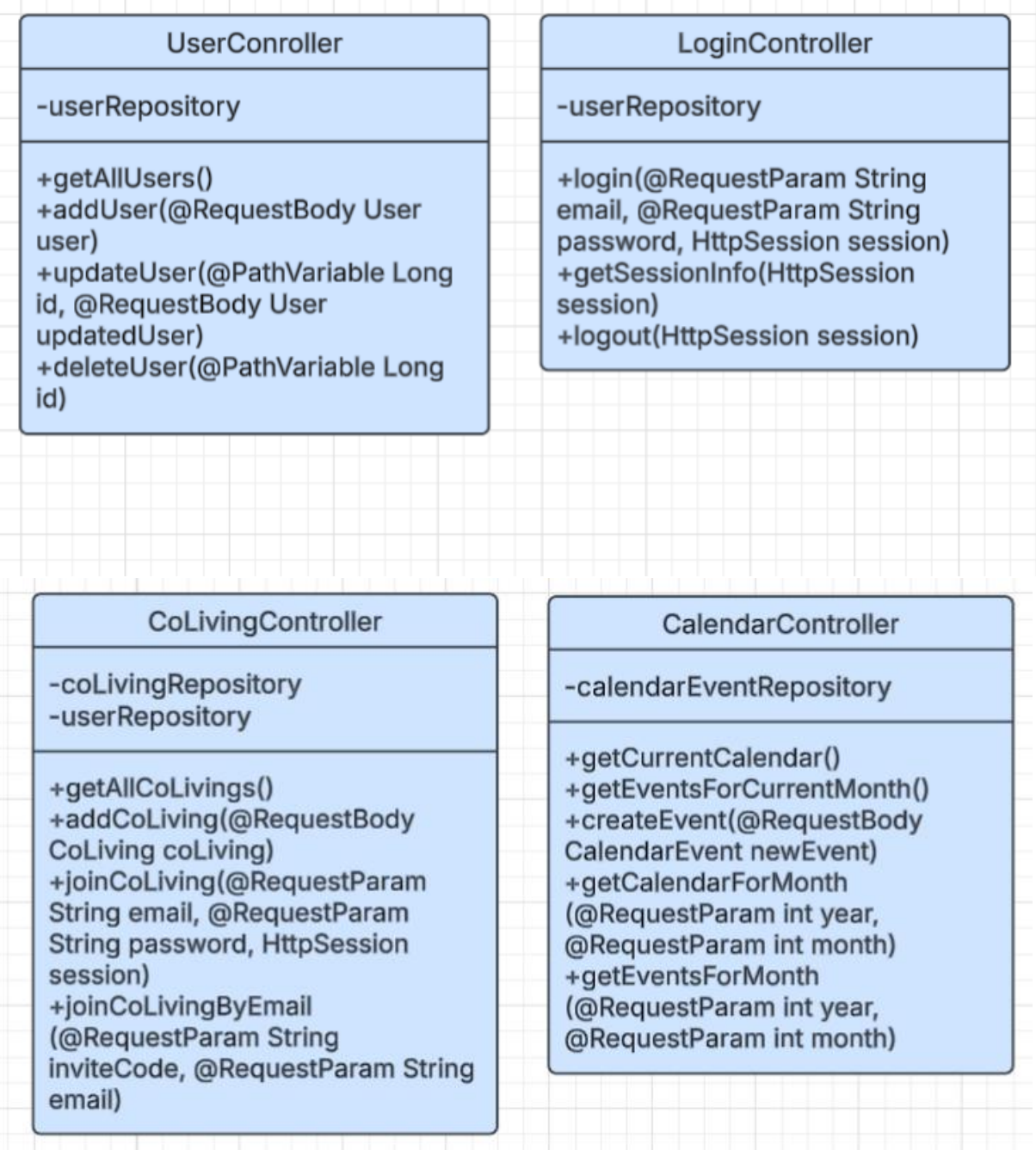
4.2 UML: diagrammer

4.2.1 UML

Figur 6 og 7 viser UML klassediagrammene for modeller og kontrollere i prosjektet, med tilhørende funksjoner.



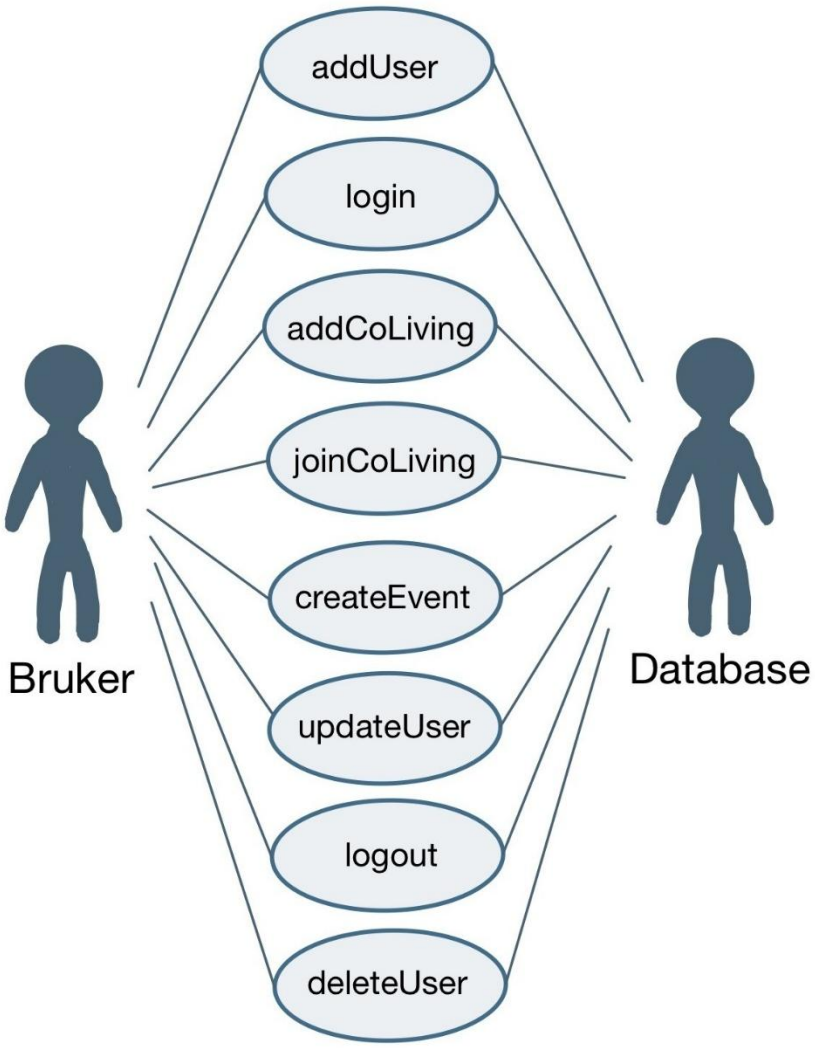
Figur 6 - Klassediagram (UML) for modeller



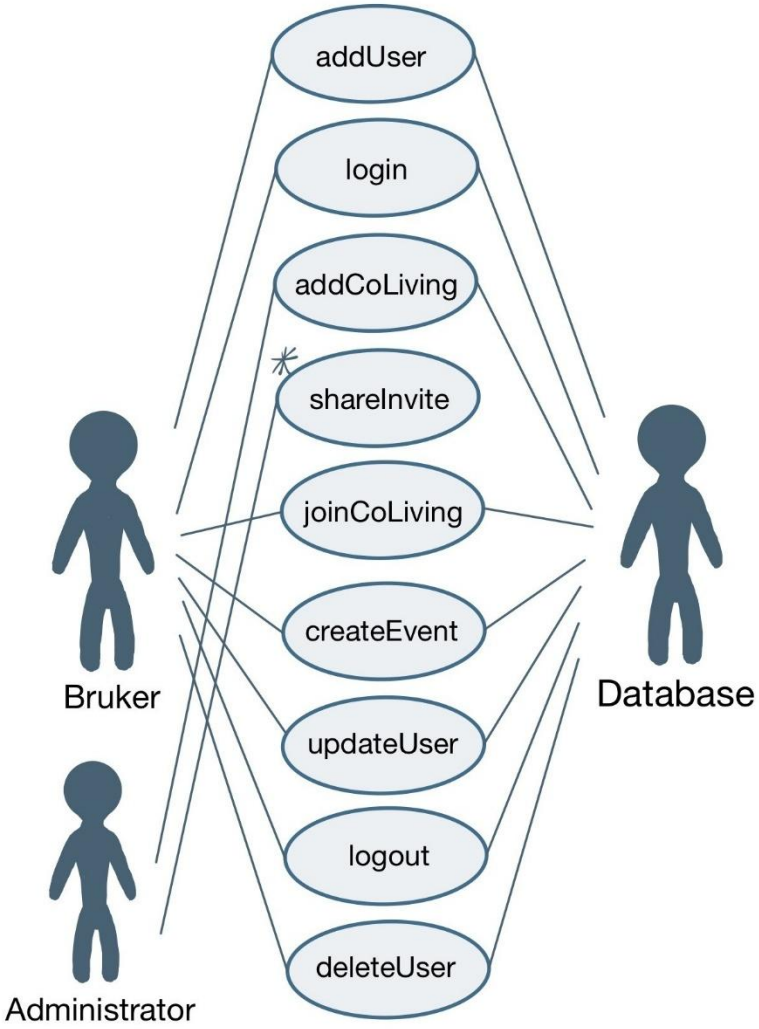
Figur 7 - Klassediagram (UML) for kontrollere/API

4.2.2 Use Case

Figur 8 og 9 viser Use-Case for webapplikasjonen. Figur 8 viser slik det er nå, og figur 9 viser slik det er ønskelig at det skal være.



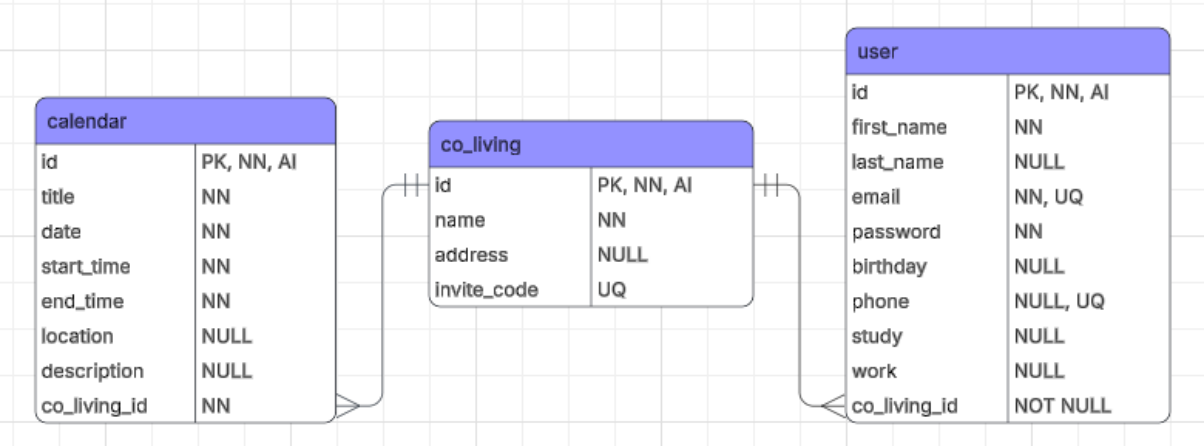
Figur 8 - Use-case for den faktiske webapplikasjonen per nå



Figur 9 - Use-case for webapplikasjonen med administrator

5 Database

Databasediagrammet på figur 10 viser tabellene i databasen og relasjonen mellom dem. Det er tre tabeller. «co_living» er «hoved»-tabellen som alt annet knyttes til med foreign keys. «user» holder all informasjon om brukere. «calendar» holder informasjon om hendelser. Her også tilknyttet «co_living» for senere å kunne være unik for hvert kollektiv i stedet for en kjempestor kalender for alle kollektiver.



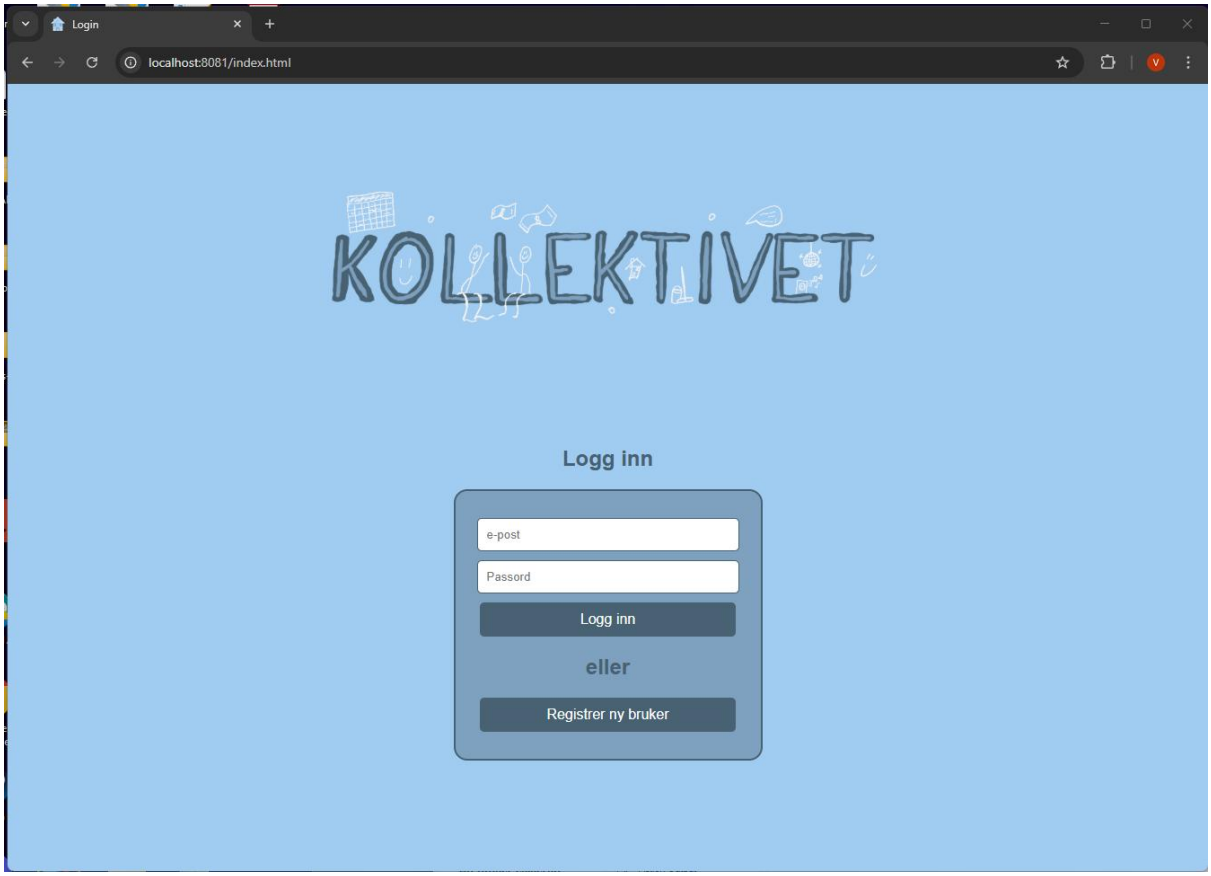
Figur 10 - Databasediagram (ERD)

6 Webapplikasjonen

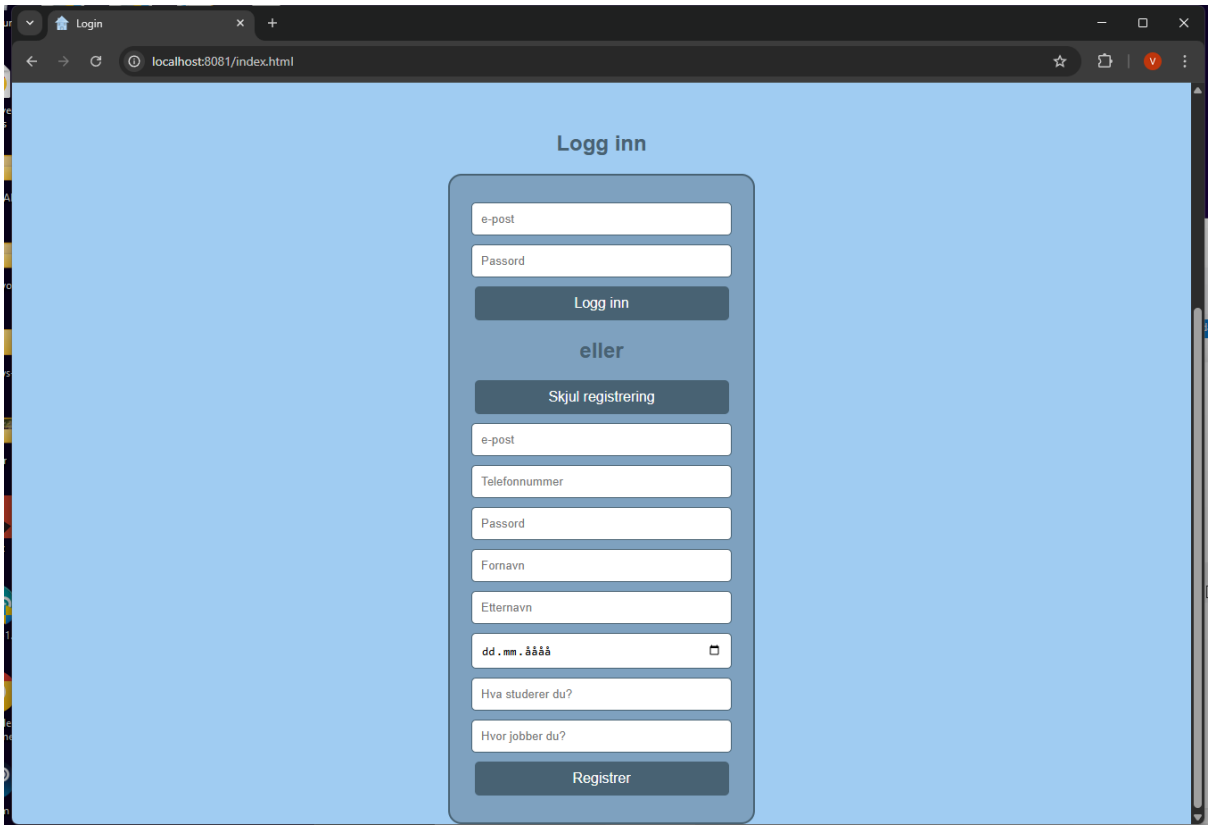
Slik webapplikasjonen er nå, har brukeren 9 funksjoner å velge mellom: Registrere bruker, Logge inn, Opprette et nytt kollektiv, Delta i et kollektiv, Opprette en hendelse i kalenderen, Oppdatere profilen sin, Logge ut og Slette bruker.

6.1 Index.html

Index.html er der man «starter». Her vises en stor logo og en boks med input-felter. Det er en knapp som bruker JavaScript for å utvide boksen slik at input-feltene for å registrere bruker kommer til synet. Figur 11 viser et skjermbilde av login-siden (index.html) ved start. Figur 12 viser login-siden med registreringsskjema.



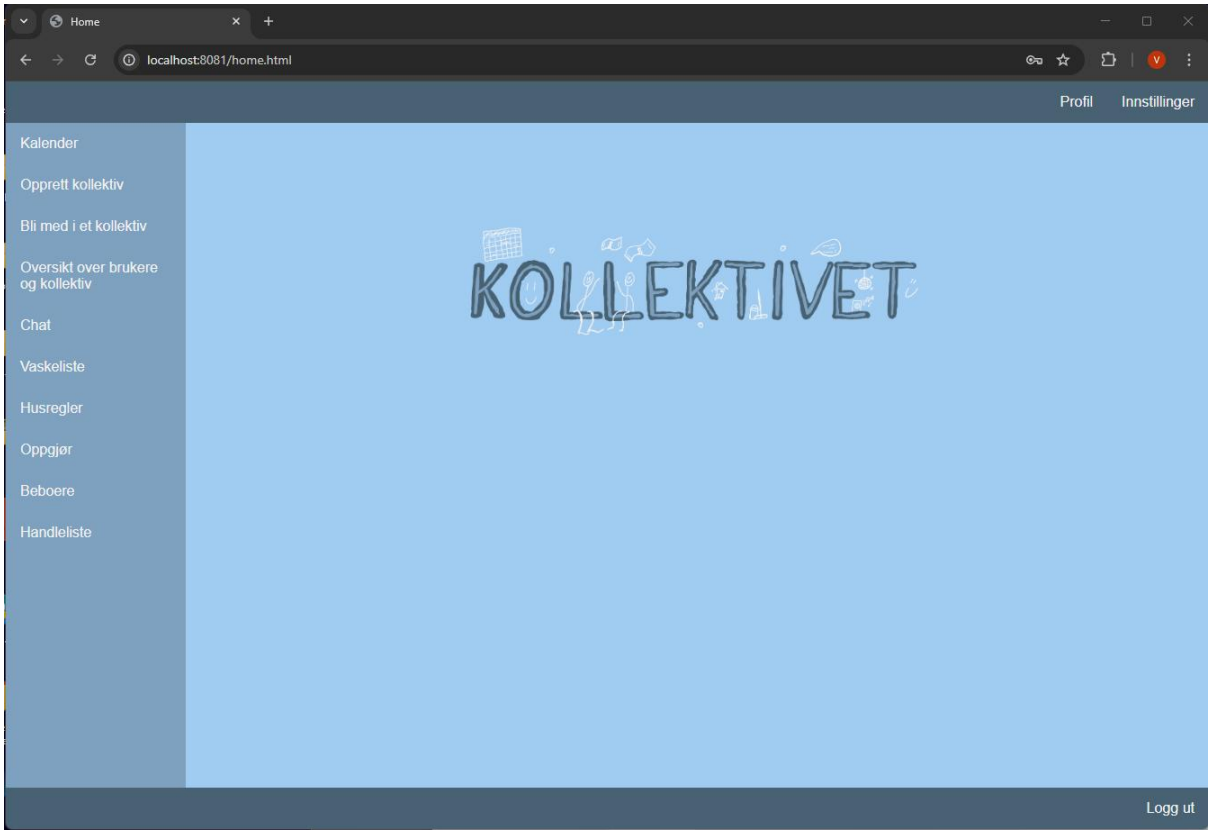
Figur 11 - Skjermbilde av index.html login-delen



Figur 12 - skjermbilde av index.html register-delen

6.2 home.html

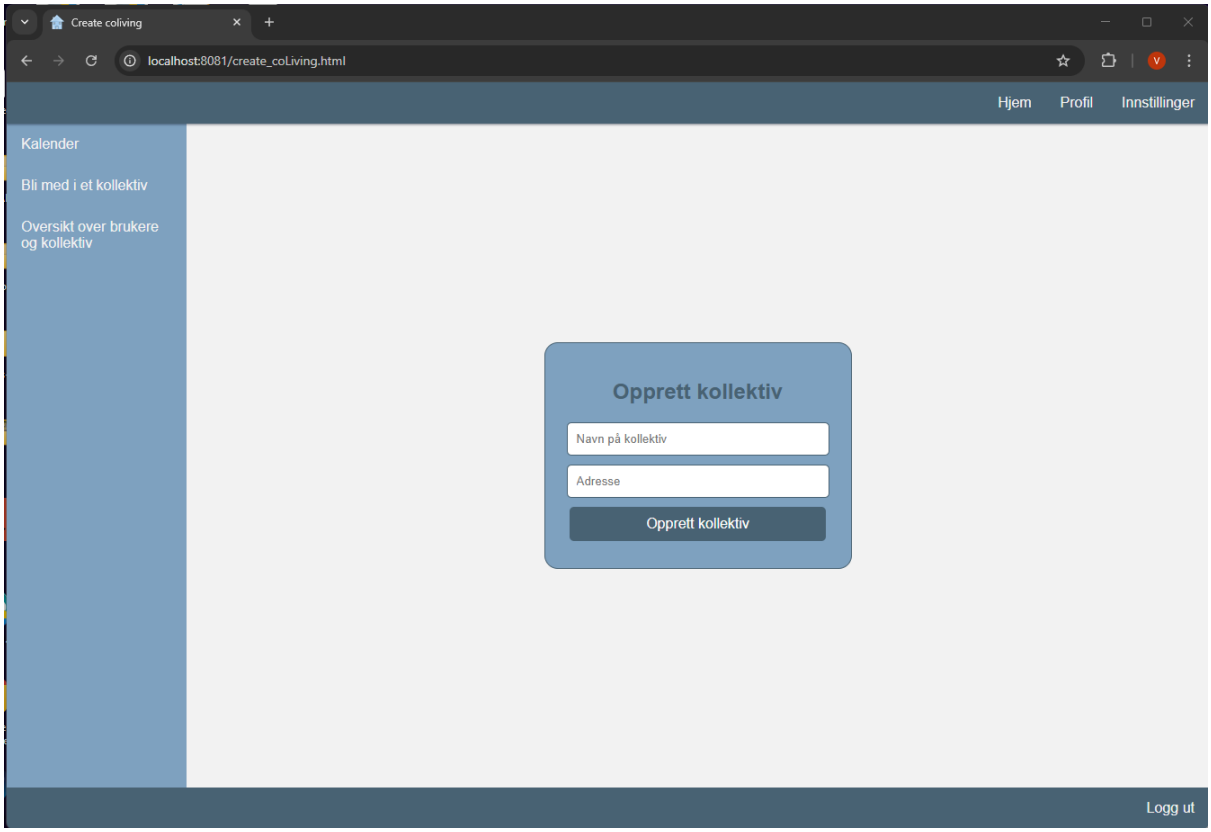
Hjemsiden er veldig simpel og bruker stort sett bare HTML og CSS, for utenom litt JavaScript for å logge ut. På hjemsiden er det flere lenker til andre sider i sidemenyen enn for resten av filene i prosjektet. Dette er kun for å vise hvilke andre filer og funksjoner som kunne vært relevant å ha med. Figur 13 viser skjermbilde av hjemsiden (home.html).



Figur 13 - skjermbilde av home.html

6.3 create_coLiving.html

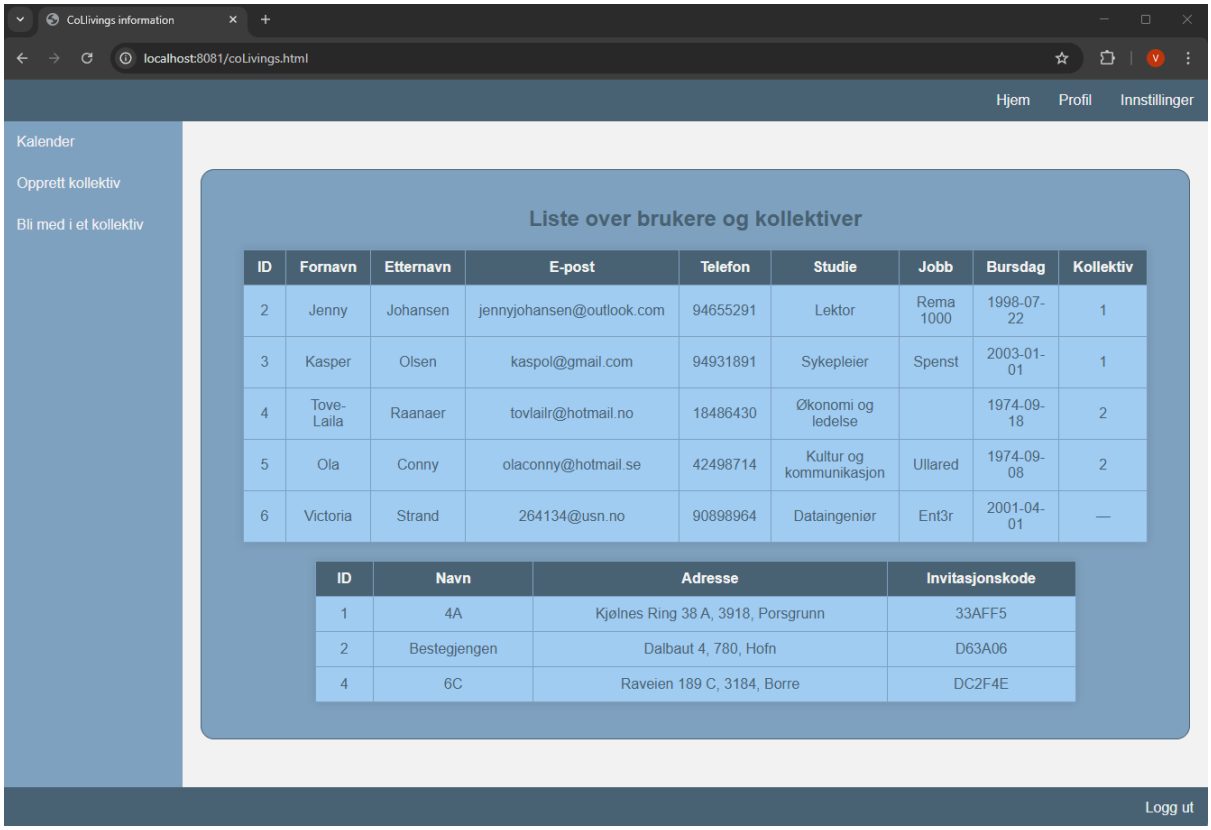
Siden for å opprette et kollektiv består av samme layouten med en toppmeny, sidemeny og «logg ut»-knapp nederst, og har et inputfelt for å skrive inn navn og adresse på det nye kollektivet. Så sant det ikke oppstår en feil blir det lagt til et nytt kollektiv i databasen når man trykker på «Opprett kollektiv»-knappen. Ved oppretting genereres det automatisk en unik invitasjonskode som også lagres i databasen uten at det synes. Figur 14 viser skjermbilde av create_coLiving.html.



Figur 14 - skjermbilde av create_coLiving.html

6.4 coLivings.html

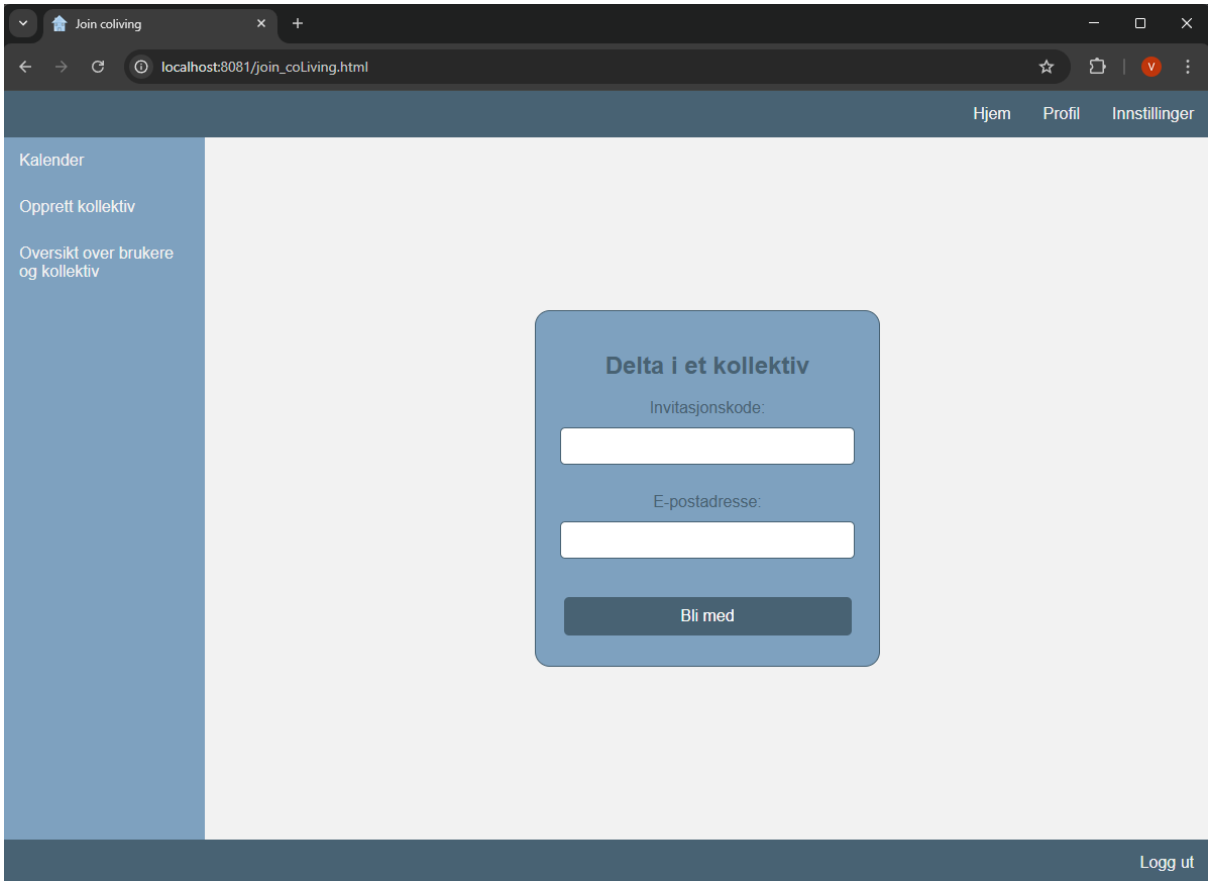
coLivings.html er en egen side med oversikt over brukere og kollektiver. Her hentes data fra databasen og presenteres på en oversiktlig måte. I senere utvikling ville denne siden bare vært tilgjengelig for administratorer. Det ville også vært gjort noen endringer slik at ikke ALL informasjon vises, for eksempel passord. Figur 15 viser skjermbilde av coLivings.html.



Figur 15 - skjermbilde av coLivings.html

6.5 join_coLiving.html

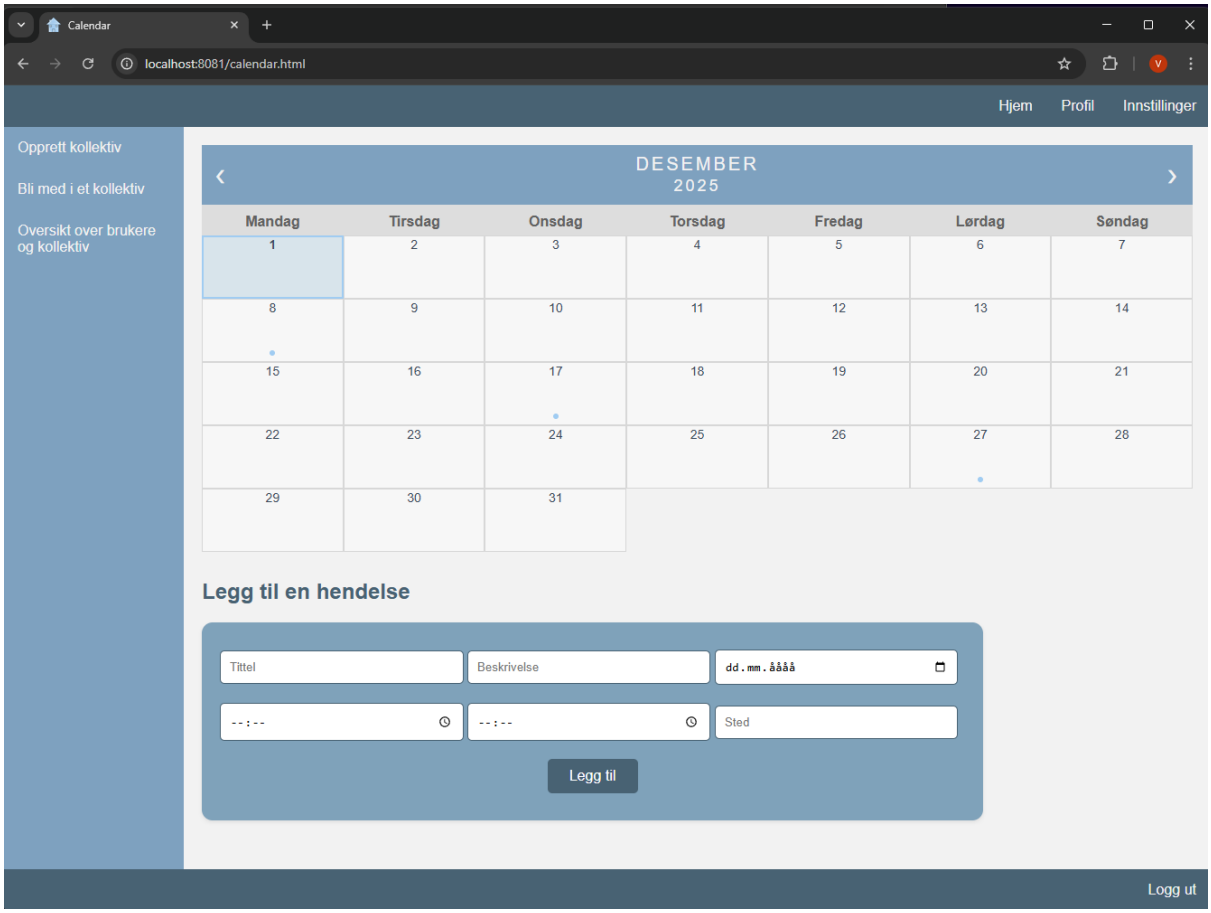
Siden for å delta i et kollektiv består av et inputfelt som tar inn den unike invitasjonskoden for ønsket kollektiv og e-posten til brukeren som skal legges til i det kollektivet. Da det er en pågående sesjon kunne en også brukt det for å slippe å skrive inn e-post manuelt. Denne funksjonaliteten brukes i profile.html og settings.html. Figur 16 viser skjermbilde av join_coLiving.html.



Figur 16 - skjermbilde av join_coLiving.html

6.6 calendar.html

Kalender-siden viser en kalender og et input-felt for å opprette hendelser. Dette blir lagret i databasen. Figur 17 viser et skjermbilde av calendar.html.



Figur 17 - skjermbilde av calendar.html

Kalendersiden er den delen av prosjektet som har krevd mest formatering og logikk. Her er store deler av koden hentet fra eller laget i samarbeid med KI-modellen ChatGPT (OpenAI, 2025). Figur 18, 19 og 20 viser utsnitt fra calendar.html over scriptet som sørger for at kalenderen ser fin ut, oppdateres basert på dagens dato, og henter hendelser og markerer dem i kalenderen.

```
<script>
    const daysContainer = document.querySelector(".days");
    const monthNameEl = document.getElementById("month-name");
    const yearEl = document.getElementById("year");

    let cachedEvents = [];

    async function fetchCalendarMeta() {  Show usages  Victoria Strand
        const response = await fetch("http://localhost:8081/api/calendar/current");
        if (!response.ok) {
            throw new Error("Feil ved henting av kalender-metadata");
        }
        return await response.json();
    }

    async function fetchEvents() {  Show usages  Victoria Strand
        const response = await fetch("http://localhost:8081/api/calendar/events");
        if (!response.ok) {
            throw new Error("Feil ved henting av hendelser");
        }
        return await response.json();
    }

    function getDayFromDateString(dateStr) {  Show usages  Victoria Strand
        if (!dateStr) return null;
        const parts = dateStr.split("-");
        if (parts.length !== 3) return null;
        return parseInt(parts[2], 10);
    }

    async function renderCalendar() {  Show usages  Victoria Strand
        try {
            const [meta, events] = await Promise.all([
                fetchCalendarMeta(),
                fetchEvents()
            ]);

            cachedEvents = events;

            monthNameEl.textContent = meta.monthName;
            yearEl.textContent = meta.year;

            daysContainer.innerHTML = "";

            const offset = (meta.firstDayOfWeek + 6) % 7;
```

Figur 18 - Script for formatering og oppdatering av kalender (del 1 av 3)

```
        for (let i = 0; i < offset; i++) {
            const emptyLi = document.createElement("li");
            emptyLi.classList.add("empty");
            daysContainer.appendChild(emptyLi);
        }

        for (let day = 1; day <= meta.daysInMonth; day++) {
            const li = document.createElement("li");
            li.dataset.day = String(day);

            const daySpan = document.createElement("span");
            daySpan.classList.add("day-number");
            daySpan.textContent = day;
            li.appendChild(daySpan);

            if (day === meta.todayDayOfMonth) {
                li.classList.add("today");
            }

            const eventsForDay = events.filter(ev => getDayFromDateString(ev.date) === day);

            if (eventsForDay.length > 0) {
                li.classList.add("has-event");
            }

            daysContainer.appendChild(li);
        }

    } catch (error) {
        console.error("Klarte ikke å rendre kalender:", error);
    }
}

daysContainer.addEventListener("click", function (e) {
    const li = e.target.closest("li[data-day]");
    if (!li) return;

    const day = parseInt(li.dataset.day, 10);

    const eventsForDay = cachedEvents.filter(ev => getDayFromDateString(ev.date) === day);

    if (eventsForDay.length > 0) {
        li.classList.add("has-event");
    }
```

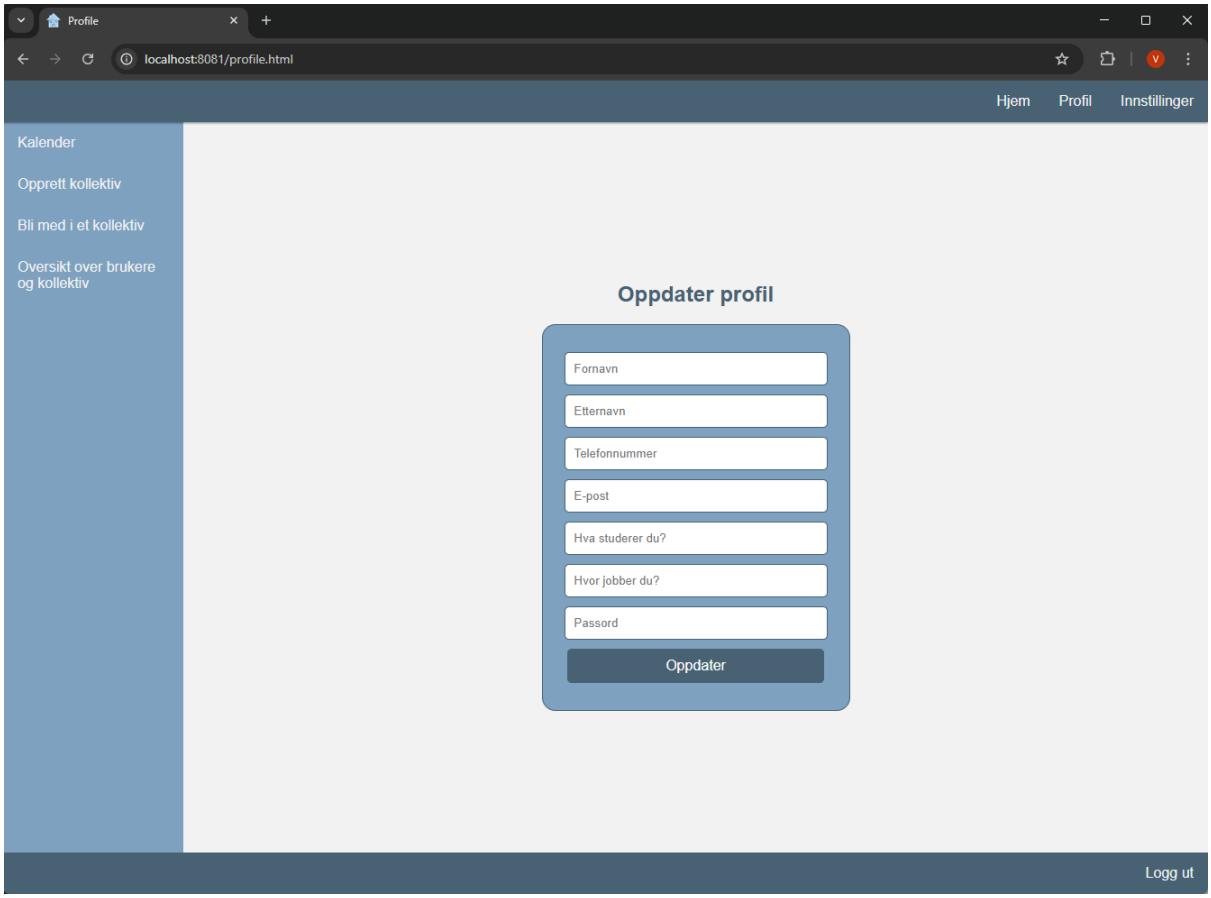
Figur 19 - Script for formatering og oppdatering av kalender (del 2 av 3)

```
        const message = eventsForDay.map(ev => {
            return (
                "Tittel: " + (ev.title ?? "") + "\n" +
                "Dato: " + (ev.date ?? "") + "\n" +
                "Start: " + (ev.startTime ?? "") + "\n" +
                "Slutt: " + (ev.endTime ?? "") + "\n" +
                "Sted: " + (ev.location ?? "") + "\n" +
                "Beskrivelse: " + (ev.description ?? "")
            );
        }).join("\n\n----------------\n\n");

        alert(message);
    });

    renderCalendar();
</script>
```

Figur 20 - Script for formatering og oppdatering av kalender (del 3 av 3)

6.7 profile.html

Profilsiden viser et input-felt der man kan endre brukerinformasjonen sin. Her brukes sesjon-funksjonalitet for å hente ut brukeren som er pålogget. Figur 21 viser et skjermbilde av profile.html.



Figur 21 - skjermbilde av profile.html

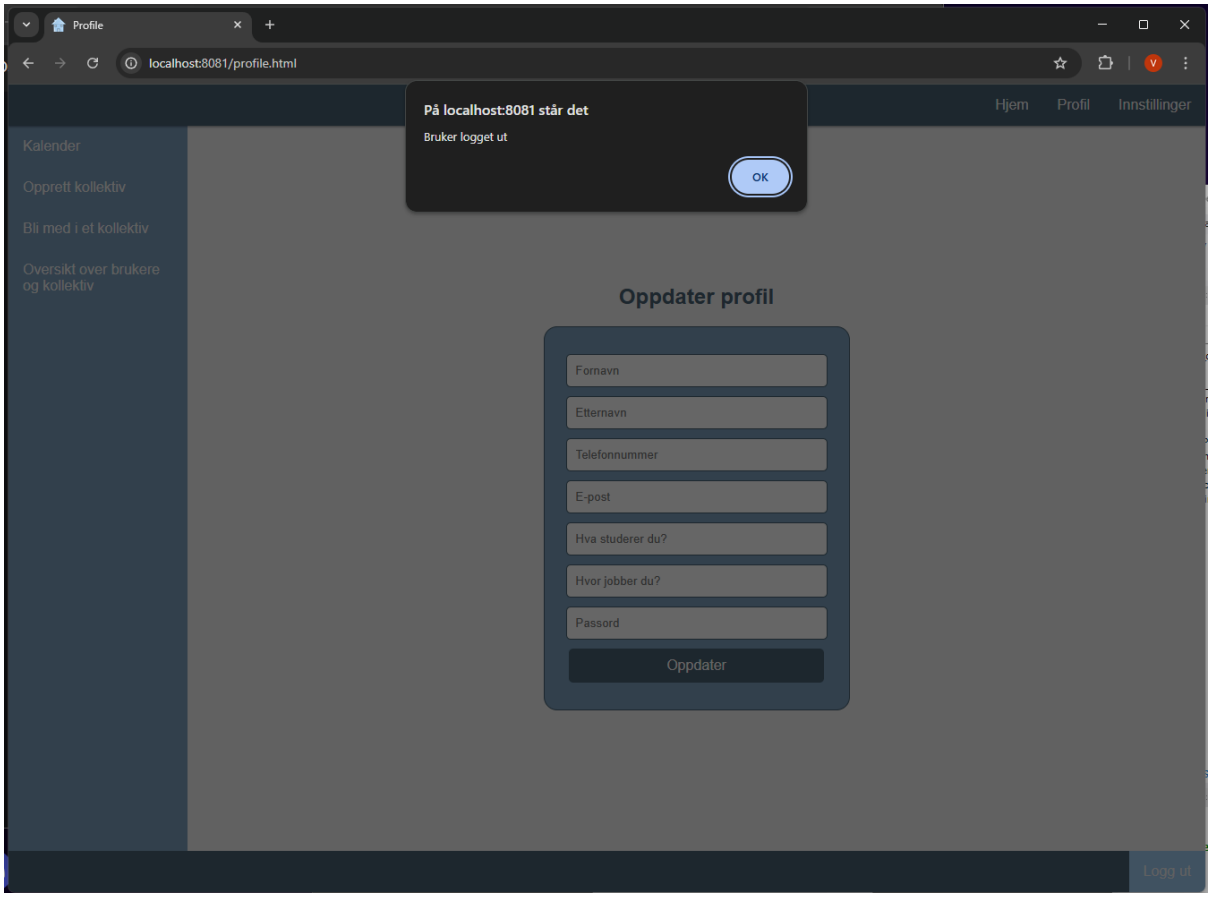
6.8 logout

Logout er ikke en egen side, men et script som ligger i tilnærmet alle de andre filene. Scriptet avslutter nåværende sesjon og sender brukeren tilbake til login-siden (index.html). Figur 22 viser et skjermbilde av dette scriptet.

```
<script>
    document.getElementById("logoutBtn").addEventListener("click", async (e) => {
        e.preventDefault();

        try {
            const response = await fetch("http://localhost:8081/api/login_out/logout", {
                method: "POST",
                credentials: "include"
            });

            const result = await response.text();
            alert(result);

            window.location.href = "index.html";

        } catch (err) {
            console.error("Feil:", err);
            alert("Kunne ikke logge ut");
        }
    });
</script>
```

Figur 22 - Skjermbilde av script for å logge ut/avslutte sesjon

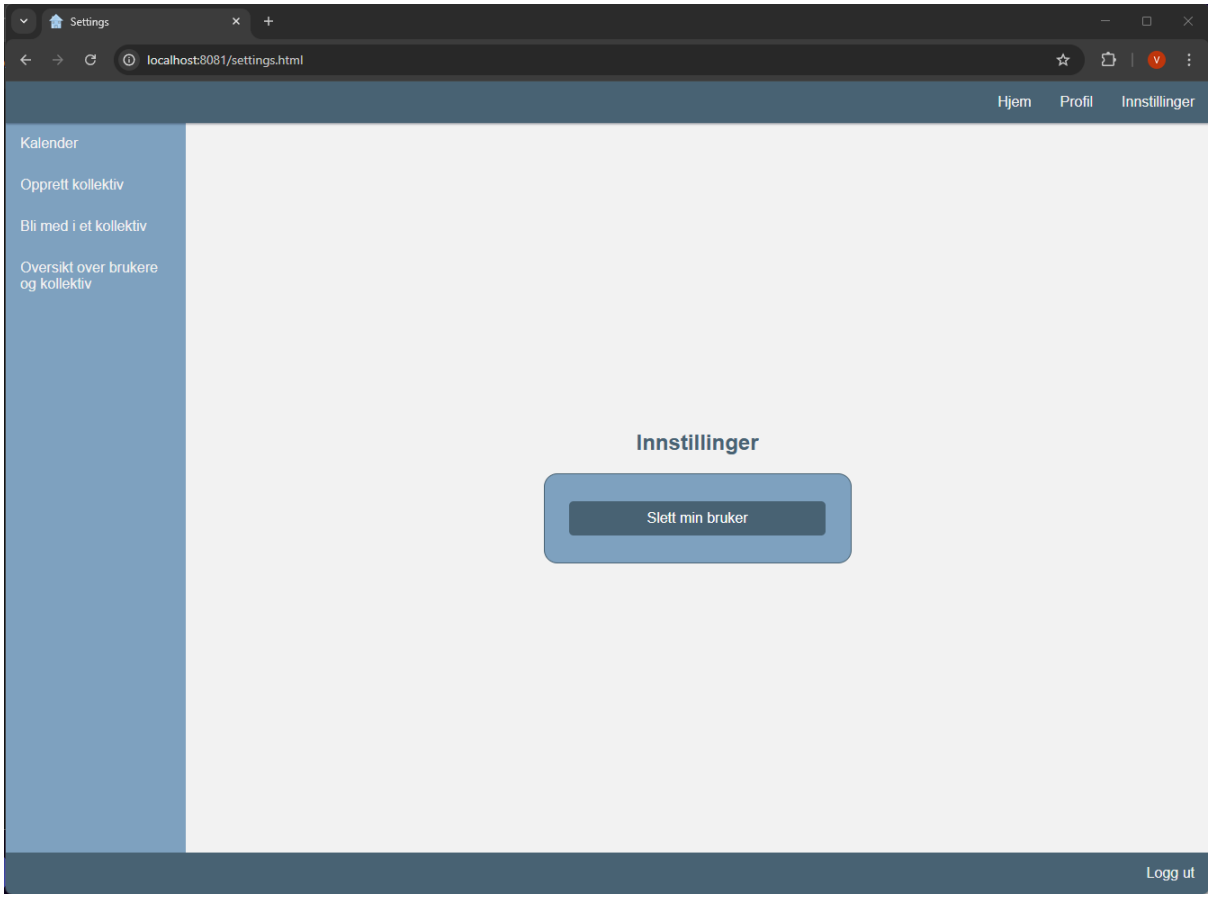
Figur 23 viser et skjermbilde av hvordan det ser ut når man logger ut.



Figur 23 - skjermbilde av bekreftet utlogging

6.9 settings.html

Siden for innstillinger består for nå kun av en knapp som sletter brukeren som er logget inn. Ved sletting avsluttes sesjonen og man blir sendt tilbake til login-siden (index.html), og brukeren fjernes fra databasen. Figur 24 viser et skjermbilde av settings.html.



Figur 24 - skjermbilde av settings.html

7 Diskusjon/oppsummering

KOLLEKTIVET er per nå en fungerende webapplikasjon, men med mange begrensninger. Den bærer stort preg av å være under utvikling. Dette er ikke uforventet basert på ferdigheter innenfor webutvikling på forhånd, tidsramme, høye ambisjoner og en idé som er svært omfattende.

I videre utvikling ville det vært nødvendig å endre prosjektet slik at det er to versjoner; én for administratorer og én for brukere. Det ville også vært nødvendig å begrense tilgang slik at kun de administratorene og brukerne tilknyttet et gitt kollektiv kan se og endre informasjon for det spesifikke kollektivet.

Aller viktigst av alt for en velfungerende webapplikasjon som håndterer brukerdata og passord er sikkerhet. Per nå er det ikke satt inn noen tiltak spesifikt rettet mot dette. Dette er ikke førsteprioritet for testing, men det er det absolutt viktigste dersom «KOLLEKTIVET» faktisk skulle blitt brukt i virkeligheten. Første steg her ville vært å bruke en hash-funksjon for å unngå å lagre passord direkte i databasen.

Det ville også vært ønskelig å legge til flere sider og funksjoner som vaskeliste med oversikt over hvem som skal vaske når, husregler, felles oversikt over økonomi, språkinnstillinger, felles chatterom, felles handleliste, oversikt over hvem som bor i kollektivet, mulighet for å forlate et kollektiv og å slette kollektiv.

Referanser

GeeksforGeeks. (2025). *Spring Boot Tutorial*. Hentet fra geeksforgeeks.org: https://www.geeksforgeeks.org/advance-java/spring-boot/

OpenAI. (2025). *ChatGPT*(2025 versjon) [Stor språkmodell]. https://chat.openai.com/

Stack Exchange Inc. (2025). *2025 Developer Survey: Most popular technologies*. Hentet fra survey.stackoverflow.co: https://survey.stackoverflow.co/2025/technology#most-popular-technologies

VMware Tanzu. (2025). *Spring makes Java productive.* Hentet fra spring.io: https://spring.io/

Figurliste