



Kristiania

# EKSAMEN

PG5602 H2025

# iOS PROGRAMMERING

WHERE THINKERS BECOME MAKERS



The world is a book, and  
those who do not travel  
only read a page.  
– St. Augustine

# INTRODUKSJON

Nå er tiden inne. Gjennom semesteret har du lært hvordan man bygger moderne apper med SwiftUI, hvordan man henter og håndterer data, og hvordan man skaper gode brukeropplevelser. Denne eksamsoppgaven gir deg muligheten til å samle alt du har lært – og bruke det til å skape noe eget.

Du skal utvikle en app med navnet **Beacon** som kombinerer kart, søker, data og design. Du får jobbe med et ekte API, vise steder på kart, legge til animasjoner og lagre informasjon lokalt. Det handler ikke bare om å få ting til å fungere – det handler om å forme en opplevelse, en app du kan være stolt av.

Tenk på dette som mer enn en eksamen. Det er en sjanse til å vise hva du kan, til å eksperimentere, og til å bygge noe som faktisk kunne vært ute i verden. Vær nysgjerrig. Vær kreativ. Og husk: du har verktøyene, nå er det opp til deg å bruke dem.

## RAMMEVILKÅR

Følgende rammebetingelser gjelder for eksamen, og det er viktig at du setter deg grundig inn i dem før du starter arbeidet. Sørg for at alle krav og retningslinjer følges nøyde gjennom hele prosessen.

### Eksamensform

Eksamen gjennomføres som en skriftlig, individuell hjemmeeksamen. Dette gir deg fleksibilitet til å jobbe i ditt eget tempo og miljø, samtidig som det forventes at alt arbeid er ditt eget.

### Kvalitet på kode

Det legges stor vekt på strukturert og ryddig programmering. Koden din skal være oversiktlig og lett å forstå, samtidig som den tilfredsstiller alle funksjonelle krav som er spesifisert i oppgaven. Tenk på at god struktur også gjør det enklere å videreføre eller feilsøke applikasjonen.

### Design og brukeropplevelse

Applikasjonen skal ha et visuelt design som følger anerkjente retningslinjer for mobilapper. Du forventes å bruke Assets-funksjonen i Xcode til å organisere og optimalisere grafiske ressurser som farger, ikoner og bilder. God design handler ikke bare om estetikk, men også om å skape en intuitiv og brukervennlig opplevelse for brukeren.

### Språkkrav

Brukergrensesnittet i appen skal være på norsk, mens all Swift-kode skal skrives på engelsk. Dette gjenspeiler profesjonell praksis og forbereder deg på reelle utviklingsprosjekter der slike krav ofte gjelder.

### **Feilhåndtering**

Det forventes at alle steder appen kan feile (eks: nettverkskall), så skal man vise en forståelig feilmelding til brukeren.

### **Krav til leveranse**

Når prosjektet er fullført, skal det kunne kompileres og kjøres direkte i iPhone-simulatoren uten endringer. Komprimer hele prosjektmappen til en ZIP-fil før du laster den opp i WISEflow. Kontroller at alle nødvendige filer er inkludert før innlevering – dobbeltsjekk dette!

Husk også å legge ved videofilen.len!

### **Presentasjon av prosjektet**

I tillegg til selve appen skal du lage en kort presentasjonsvideo der du demonstrerer applikasjonen i simulatoren. Videoen skal også inneholde en forklaring av sentrale deler av koden din. Dette gir sensor innsikt i dine valg og vurderinger, og viser at du har forstått både tekniske og designmessige aspekter ved prosjektet.

Ved å følge disse retningslinjene legger du et solid grunnlag for et vellykket eksamensprosjekt. Husk at både kreativitet og nøyaktighet teller.

## **GEOAPIFY**

Geoapify Places API lar deg søke etter steder og interessepunkter (POI) basert på kategori, geografisk område eller søkeord. Den støtter både autocomplete og detaljvisning, og gir deg tilgang til informasjon som navn, adresse, koordinater, åpningstider, kontaktinfo og mer.

Tjenesten krever en API-nøkkel, som kan hentes gratis ved registrering. Geoapify er optimalisert for rask respons, god dekning og enkel integrasjon med kartbiblioteker som MapKit, og egner seg godt til applikasjoner som krever geografisk søk og visning.

## **GJENNOMFØRING**

Applikasjonen skal integrere Geoapify Places API for å hente og vise informasjon om steder. For å få tilgang til API-et må du registrere deg på Geoapifys nettside og hente en gratis API-nøkkel.

API-et tilbyr flere endepunkter, hvorav autocomplete og place details er relevante i denne sammenhengen. Autocomplete gir dynamiske forslag basert på brukerens input og geografisk kontekst, mens place details returnerer strukturert informasjon om et valgt sted, inkludert navn, adresse, koordinater og metadata.

Dataene hentes som JSON og må dekodes og presenteres i appens grensesnitt. Løsningen skal være responsiv, håndtere feilsituasjoner, og gi brukeren en sømløs opplevelse ved søker og visning av steder.

# BRANDINGPAKKE



**Beacon**

Beacon bruker en stilisert lokasjonspinne med en stjerne i sentrum, et symbol på oppdagelse, retning og kvalitet. Ikonet er sirkulært med mørk bakgrunn og en varm oransje kantlinje, og fungerer godt både i lys og mørk modus.

Navn	Hex kode	Bruksområde
Beacon Orange	#FF9B52	Knapper og titler
Highlight Orange	#F7853E	Stjernerating og favoritter
Deep Blue	#20202A	Bakgrunn til kart, kort, ikonbase



## Overskrifter, knapper og navigasjon

Vennlig, tydelig og profesjonell – bruk sentence-case og unngå overflødig dekor

### Komponentstil

Knapper: Runde hjørner, fylt med Beacon Orange, hvit tekst.



# HOVEDLAYOUT

Appens hovedvisning består av et MapKit-kart som fyller hele skjermen. Kartet er det sentrale elementet og gir brukeren oversikt over relevante steder i området. Øvrige funksjoner legges som overlays eller navigeres via en TabView.

Nede til høyre i kartet plasseres en GPS-knapp med ikon som lar brukeren sentrere kartet på sin nåværende posisjon. Over kartet vises en kategori-velger i form av en **Picker** med segmented style. Denne lar brukeren velge mellom ulike typer steder, som restauranter, kafeer og hoteller.

Ved siden av kategori-velgeren vises en egen knapp for å hente steder fra Geoapify Places API. Brukeren må aktivt trykke på denne knappen etter å ha valgt lokasjon og kategori. Resultatene vises som pins på kartet og kan også presenteres i en listevisning.

Appen benytter en **TabView** med to faner:

- **Utforsk:** Kart, kategori-filter og henteknapp.
- **Mine steder:** Liste over lagrede severdigheter, lagret med SwiftData.

Layouten skal være enkel, oversiktlig og inspirere til utforskning. Kartet er alltid i fokus, mens interaktive elementer er plassert intuitivt og med god kontrast.

# OPPGAVE 1

I denne oppgaven skal du implementere funksjonalitet som følger hovedlayouten beskrevet over. Appens hovedvisning består av et MapKit-kart som fyller hele skjermen, med interaktive komponenter plassert som overlays.

Ved første gangs oppstart skal appen bruke en **default lokasjon**, for eksempel Oslo S (59.9111, 10.7503). Kartet skal vise denne posisjonen og være klar til å hente relevante steder. Brukeren skal kunne **veksle mellom kart- og listevisning**, der kartet viser pins for restauranter og listen viser navn og adresse.

Brukeren må selv trykke på en **egen knapp** for å hente data fra Geoapify Places API. Denne knappen skal plasseres ved siden av kategori-velgeren, som vises som en **Picker** med segmented style øverst i kartvisningen. Søket skal alltid baseres på **kartets sentrum**, og det skal vises de **10 nærmeste restaurantene**.

Kartets sentrum skal lagres i **AppStorage**. Når brukeren aktivt endrer sentrum, skal den nye posisjonen lagres. Dette gjør at appen husker hvor brukeren sist søkte, og bruker det som utgangspunkt neste gang den åpnes.

Du skal plassere komponentene slik:

- **GPS-knapp** nede til høyre i kartet
- **Kategori-velger** øverst som overlay
- **Henteknapp** ved siden av kategori-velgeren
- **Toggle mellom kart og liste** som del av hovedvisningen
- **TabView** med to faner: Utforsk og Mine steder

One's destination is  
never a place, but a new  
way of seeing things.

–Henry Miller

Fokuser på å følge layouten og arbeidskravene nøyne. Du står fritt til å forbedre visuell stil og interaksjon, men funksjonaliteten må være tydelig og korrekt.

# OPPGAVE 2

I denne oppgaven skal du utvide appens funksjonalitet slik at brukeren kan filtrere steder basert på kategori. Målet er å gi en mer relevant og personlig opplevelse, der brukeren selv velger hvilken type steder de ønsker å utforske.

Øverst i kartvisningen skal du plassere en **Picker** med *segmented style* som lar brukeren velge mellom tre kategorier: **restauranter**, **kafeer** og **hoteller**. Når brukeren endrer kategori, skal appen hente nye steder fra Geoapify Places API basert på den valgte kategorien og kartets sentrum. Det skal vises **minst 10 relevante steder** for hver kategori.

Resultatene skal oppdateres både på **kartet** (som pins) og i **listevisningen**, slik at brukeren får både visuell og tekstlig oversikt. Endringen skal skje dynamisk og gi tydelig tilbakemelding i grensesnittet.

Som en utvidelse skal du implementere en funksjon som lar brukeren finne steder i **nærheten av seg selv**. Dette innebærer å be om brukerens tillatelse til å hente posisjon, og deretter bruke denne posisjonen som sentrum for søker. Du kan gjerne bruke en egen knapp med teksten "Finn nærmeste meg" eller et GPS-ikon som signaliserer funksjonen.

## A R B E I D S K R A V :

- **Picker** med segmented style for kategori (restauranter, kafeer, hoteller)
- Hent nye steder fra Geoapify når kategori endres
- Vis minst 10 steder for valgt kategori
- Oppdater både kart og liste dynamisk
- Implementer "Finn nærmeste meg" med posisjonstilgang

## O P P G A V E 3

I denne oppgaven skal du implementere en detaljvisning som vises når brukeren trykker på en pins i kartet eller listen. Visningen skal presentere relevant informasjon om stedet, og samtidig gi en visuell opplevelse som varierer basert på hvilken kategori stedet tilhører. Dette gir en mer levende brukeropplevelse.

Detaljvisningen skal presenteres enten via **NavigationStack** eller som en **Sheet**, og inneholde følgende informasjon:

- Navn på stedet
- Adresse
- Koordinater
- Hvis tilgjengelig: åpningstider, kategori og telefonnummer
- En knapp som åpner stedet i Apple Maps

I tillegg skal du implementere en **kategori-spesifikk animasjon** øverst i visningen. Hver kategori har sitt eget emoji-symbol og animasjonsstil:

### 🍽 RESTAURANT

- Vis 🍽 øverst i detaljvisningen
- Når visningen åpnes, skal emojien rotere 360°
- Animasjonen skal vare i 1 sekund
- Bruk `.easeInOut` som timing-funksjon
- 

### ☕ KAFÉ

- Vis ☕ øverst i detaljvisningen
- Tre "damp-partikler" skal animeres oppover fra koppen
- Bruk `Text("wat")` eller `Circle()` for partikler

- Hver partikkel skal fade ut mens den beveger seg oppover (opacity fra 1.0 til 0.0)
- Animasjonen skal gjentas kontinuerlig
- Bruk `.easeOut` som timing-funksjon

## HOTELL

- Vis  øverst i detaljvisningen
- Emojien skal “bounce” når visningen åpnes
- Bruk `.spring()` med bounce-effekt
- Start fra `scale = 0.5` og animer til `scale = 1.0`

## OPPGAVE 4

I denne oppgaven skal du implementere en vurderingsfunksjon som lar brukeren gi tilbakemelding på steder de har besøkt eller ønsker å anbefale. Vurderingene skal lagres lokalt med SwiftData og bevares mellom app-kjøringer. Dette gir appen et personlig preg og gir brukeren mulighet til å bygge opp sin egen historikk.

Fra detaljvisningen skal brukeren kunne gi en vurdering fra **1 til 5 stjerner**. I stedet for å vise fem separate stjerner, kan du bruke **én stjerne som fylles gradvis** for å indikere vurderingsnivået. Fyllingsgraden skal gjenspeile gjennomsnittet av alle vurderinger for stedet:

- 1 stjerne = 20 % fylt
- 5 stjerner = 100 % fylt

Brukeren skal kunne gi flere vurderinger til samme sted over tid. Hver vurdering skal lagres som en **egen oppføring** i SwiftData.

Appen skal beregne **gjennomsnittlig vurdering** for hvert sted og vise dette som en fylt stjerne. Rund av til nærmeste hele stjerne Dersom et sted ikke har noen vurderinger, skal det ikke vises noen stjerne. Stjernene skal vises på kartet og i listen.

## OPPGAVE 5

I denne oppgaven skal du utvikle en fleksibel og brukervennlig søkefunksjon som lar brukeren finne steder basert på tekst, kategori, radius og sortering. Målet er å gi brukeren full kontroll over hvilke steder som vises, og hvordan de presenteres – både i kart og liste.

Appen skal inneholde et **søkefelt** der brukeren kan skrive inn navn eller nøkkelord for å finne spesifikke steder. Søk skal skje i **real-time**, og resultatene skal oppdateres mens brukeren skriver. For å unngå overbelastning av API-et, må du implementere **debounce** med minimum 300 ms forsinkelse før søkeret triggges.

Brukeren skal også kunne justere **søkeradius** med en **Slider**, fra 1 til 10 km. Radiusen skal vises tydelig og brukes som parameter i API-kallet. Standardverdien kan være 5 km fra Oslo S (59.9111, 10.7503), eller fra kartets nåværende sentrum og skal lagres ved hjelp av AppStorage. Søket skal kunne kombineres med **kategori-filteret** fra Oppgave 2 (restauranter, kafeer, hoteller). I tillegg skal brukeren kunne velge **sorteringsrekkefølge**:

- Avstand (nærmetest først)
- Rating (høyest først)
- Alfabetisk (A–Å)

Brukeren skal også kunne velge om søker skal gjelde **alle steder**, eller kun **lagrede favoritter**. Favoritter lagres med SwiftData og skal kunne filtreres separat.

Not all those who  
wander are lost.” – J.R.R.  
Tolkien

Det skal være en **clear-knapp** som nullstiller søkefelt, radius, kategori og sortering, og gjenoppretter standardvisning.

## A R B E I D S K R A V

- Real-time søk med debounce (minimum 300 ms)
- Loading state (**ProgressView**) mens søker pågår
- Håndter scenarioer der ingen resultater finnes (f.eks. vis “Ingen treff” med ikon eller emoji)
- Bruk SwiftData til å lagre og hente favoritter
- Implementer **pull-to-refresh** i listevisningen for å hente nye data manuelt

# O P P G A V E 6

Innspill et maksimum 5-minutters skjermopptak hvor du gjennomgår koden din. Her skal du vise koden i videoen mens du snakker.

## A R B E I D S K R A V :



**API-integrasjon :** Vis hvordan data flyter fra Geoapify API til UI. Forklar hvordan du parser JSON og håndterer async/await.

**SwiftData:** Forklar hvordan rating lagres og hentes. Vis modellen og hvordan du beregner gjennomsnitt.

**Animasjoner:** Velg én animasjon og forklar linje for linje hvordan den fungerer.

**Utfordringer:** Hva var vanskeligst? Hvordan løste du det?

Tips: Bruk QuickTime Player (File | New Screen Recording) eller Cmd+Shift+5 på Mac. Snakk mens du viser koden. Legg ved filen i mappen sammen med koden.





Kristiania

# EKSAMEN

PG5602 H2025

# iOS PROGRAMMERING

WHERE THINKERS BECOME MAKERS



The world is a book, and  
those who do not travel  
only read a page.  
– St. Augustine

# INTRODUKSJON

No er tida inne. Gjennom semesteret har du lært korleis ein byggjer moderne appar med SwiftUI, korleis ein hentar og handterer data, og korleis ein skaper gode brukaropplevelingar. Denne eksamensoppgåva gjev deg høvet til å samle alt du har lært – og bruke det til å skape noko eige.

Du skal utvikle ein app med namnet **Beacon** som kombinerer kart, søk, data og design. Du får jobbe med eit ekte API, vise stader på kart, leggje til animasjonar og lagre informasjon lokalt. Det handlar ikkje berre om å få ting til å fungere – det handlar om å forme ei oppleveling, ein app du kan vere stolt av.

Tenk på dette som meir enn ein eksamen. Det er ein sjanse til å vise kva du kan, til å eksperimentere, og til å byggje noko som faktisk kunne vore ute i verda. Ver nysgjerrig. Ver kreativ. Og hugs: du har verktøya, no er det opp til deg å bruke dei.

## RAMMEVILKÅR

Følgjande rammevilkår gjeld for eksamen, og det er viktig at du set deg grundig inn i dei før du startar arbeidet. Sørg for at du nøye følgjer alle krav og retningslinjer gjennom heile prosessen.

### Eksamensform

Eksamen blir gjennomført som ein skriftleg, individuell heimeeksamen. Dette gjev deg fleksibilitet til å jobbe i ditt eige tempo og miljø, samtidig som det er forventa at alt arbeid er ditt eige.

### Kvalitet på kode

Det blir lagt stor vekt på strukturert og ryddig programmering. Koden din skal vere oversiktleg og lett å forstå, samtidig som han tilfredsstiller alle funksjonelle krav som er spesifiserte i oppgåva. Tenk på at god struktur også gjer det enklare å vidareutvikle eller feilsøkje applikasjonen.

### Design og brukaroppleveling

Applikasjonen skal ha eit visuelt design som følgjer anerkjende retningslinjer for mobilappar. Du er forventa å bruke Assets-funksjonen i Xcode til å organisere og optimalisere grafiske ressursar som fargar, ikon og bilete. God design handlar ikkje berre om estetikk, men også om å skape ei intuitiv og brukarvenleg oppleveling for brukaren.

### Språkkrav

Brukargrensesnittet i appen skal vere på norsk, medan all Swift-kode skal skrivast på engelsk. Dette speglar profesjonell praksis og førebur deg på reelle utviklingsprosjekt der slike krav ofte gjeld.

### **Feilhandtering**

Det er forventa at alle stader appen kan feile (eks: nettverks-kall), så skal ein vise ei forståeleg feilmelding til brukaren.

### **Krav til leveranse**

Når prosjektet er fullført, skal det kunne kompilerast og køyrast direkte i iPhone-simulatoren utan endringar. Komprimer heile prosjektmappa til ei ZIP-fil før du lastar henne opp i WISEflow. Kontroller at alle nødvendige filer er inkluderte før innlevering – dobbeltsjekk dette!

Hugs også å leggje ved videofila.len!

### **Presentasjon av prosjektet**

I tillegg til sjølve appen skal du lage ein kort presentasjonsvideo der du demonstrerer applikasjonen i simulatoren. Videoen skal også innehalde ei forklaring av sentrale delar av koden din. Dette gjev sensor innsikt i dine val og vurderingar, og viser at du har forstått både tekniske og designmessige aspekt ved prosjektet.

Ved å følgje desse retningslinjene legg du eit solid grunnlag for eit vellukka eksamensprosjekt. Hugs at både kreativitet og nøyaktighet tel.

## **GEOAPIFY**

Geoapify Places API let deg søkje etter stader og interessepunkt (POI) basert på kategori, geografisk område eller søkjeord. Den støttar både autocomplete og detaljvisning, og gjev deg tilgang til informasjon som namn, adresse, koordinatar, opningstider, kontaktinfo og meir.

Tenesta krev ein API-nøkkel, som kan hentast gratis ved registrering. Geoapify er optimalisert for rask respons, god dekning og enkel integrasjon med kartbibliotek som MapKit, og eignar seg godt til applikasjonar som krev geografisk søk og visning.

## **GJENNOMFØRING**

Applikasjonen skal integrere Geoapify Places API for å hente og vise informasjon om stader. For å få tilgang til API-et må du registrere deg på nettsida til Geoapify og hente ein gratis API-nøkkel.

API-et tilbyd fleire endepunkt, og autocomplete og place details er relevante i denne samanhengen. Autocomplete gjev dynamiske forslag basert på inputen til brukaren og geografisk kontekst, medan place details returnerer strukturert informasjon om ein vald stad, inkludert namn, adresse, koordinatar og metadata.

Dataa blir henta som JSON og må dekodast og presenterast i grensesnittet til appen. Løysinga skal vere responsiv, handtere feilsituasjonar, og gje brukaren ei saumlaus oppleving ved søk og vising av stader.



**Beacon**

# BRANDINGPAKKE

Beacon brukar ein stilisert lokasjonspinne med ei stjerne i sentrum, eit symbol på oppdaging, retning og kvalitet. Ikonet er sirkulaert med mørk bakgrunn og ei varm oransje kantlinje, og fungerer godt både i lys og mørk modus.

Namn	Hex kode	Bruksområde
Beacon Orange	#FF9B52	Knappar og titlar
Highlight Orange	#F7853E	Stjernerating og favorittar
Deep Blue	#20202A	Bakgrunn til kart, kort, ikonbase

## Overskrifter, knappar og navigasjon

Venleg, tydeleg og profesjonell – bruk sentence-case og unngå overflødig dekor

## Komponentstil

Knappar: Runde hjørne, fylte med Beacon Orange, kvit tekst.

# HOVUDLAYOUT

Appen si hovudvising består av eit Mapkit-kart som fyller heile skjermen. Kartet er det sentrale elementet og gjev brukaren oversikt over relevante stader i området. Andre funksjonar blir lagde som overlays eller blir navigerte via ein TabView.

Nede til høgre i kartet blir ein GPS-knapp plassert med ikon som let brukaren sentrere kartet på sin noverande posisjon. Over kartet er det vist ein kategori-veljar i form av ein **Picker** med segmented style. Denne let brukaren velje mellom ulike typar stader, som restaurantar, kafear og hotell.

Ved sidan av kategori-veljaren blir vist ein eigen knapp for å hente stader frå Geoapify Places API. Brukaren må aktivt trykkje på denne knappen etter å ha valt lokasjon og kategori. Resultata blir viste som pins på kartet og kan også presenterast i ei listevising.

Appen nyttar ein **TabView** med to fanar:

- **Utforsk:** Kart, kategori-filter og henteknapp.
- **Stadene mine:** Liste over lagra attraksjonar, lagra med SwiftData.

Layouten skal vere enkel, oversiktleg og inspirere til utforskning. Kartet er alltid i fokus, medan interaktive element er plasserte intuitivt og med god kontrast.

# OPPGÅVE 1

I denne oppgåva skal du implementere funksjonalitet som følgjer hovedlayouten skildra over. Appen si hovedvising består av eit Mapkit-kart som fyller heile skjermen, med interaktive komponentar plasserte som overlays.

Ved første gongs oppstart skal appen bruke ein **default lokasjon**, til dømes Oslo S (59.9111, 10.7503). Kartet skal vise denne posisjonen og vere klar til å hente relevante stader. Brukaren skal kunne **veksle mellom kart- og listevising**, der kartet viser pins for restaurantar og lista viser namn og adresse.

Brukaren må sjølv trykkje på ein **eigen knapp** for å hente data frå Geoapify Places API. Denne knappen skal plasserast ved sidan av kategori-veljaren, som blir vist som ein **Picker** med segmented style øvst i kartvisinga. Søket skal alltid baserast på **sentrumet på kartet**, og det skal vise dei **10 nærmeste restaurantane**.

Sentrumet på kartet skal lagrast i **AppStorage**. Når brukaren aktivt endrar sentrum, skal den nye posisjonen lagrast. Dette gjer at appen hugsar kvar brukaren sist søkte, og brukar det som utgangspunkt neste gong han blir opna.

Du skal plassere komponentane slik:

- **GPS-knapp** nede til høgre i kartet
- **Kategori-veljar** øvst som overlay
- **Henteknapp** ved sidan av kategori-veljaren
- **Toggle mellom kart og liste** som del av hovedvisinga
- **TabView** med to faner: Utforsk og Mine Stader

Fokus på å følge layouten og arbeidskrava nøyte. Du står fritt til å forbetre visuell stil og interaksjon, men funksjonaliteten må vere tydeleg og korrekt.

One's destination is never a place, but a new way of seeing things.

–Henry Miller

## OPPGÅVE 2

I denne oppgåva skal du utvide funksjonaliteten til appen slik at brukaren kan filtrere stader basert på kategori. Målet er å gje ei meir relevant og personleg oppleveling, der brukaren sjølv vel kva type stader dei ønskjer å utforske.

Øvst i kartvisninga skal du plassere ein **Picker** med *segmented style* som let brukaren velje mellom tre kategoriar: **restaurantar, kafear og hotell**. Når brukaren endrar kategori, skal appen hente nye stader frå Geoapify Places API basert på den valde kategorien og sentrumet på kartet. Det skal visast **minst 10 relevante stader** for kvar kategori.

Resultata skal oppdaterast både på **kartet** (som pins) og i **listevisinga**, slik at brukaren får både visuell og tekstleg oversikt. Endringa skal skje dynamisk og gje tydeleg tilbakemelding i grensesnittet.

Som ei utviding skal du implementere ein funksjon som let brukaren finne stader i **nærleiken av seg sjølv**. Dette inneber å be om løyve frå brukaren til å hente posisjon, og deretter bruke denne posisjonen som sentrum for søker.

Du kan gjerne bruke ein eigen knapp med teksten "Finn nærmeste meg" eller eit GPS-ikon som signaliserer funksjonen.

## ARBEIDSKRAV:

- Picker med segmented style for kategori (restaurantar, kafear, hotell)
- Hent nye stader frå Geoapify når kategori blir endra
- Vis minst 10 stader for vald kategori
- Oppdater både kart og liste dynamisk
- Implementer "Finn nærmeste meg" med posisjonstilgang

# OPPGÅVE 3

I denne oppgåva skal du implementere ei detaljvising som blir vist når brukaren trykkjer på ein pins i kartet eller lista. Visinga skal presentere relevant informasjon om staden, og samtidig gje ei visuell oppleving som varierer basert på kva kategori staden tilhører. Dette gjev ei meir levande brukaroppleving.

Detaljvisinga skal presenterast anten via **NavigationStack** eller som ein **Sheet**, og innehalde følgjande informasjon:

- Namn på staden
- Adresse
- Koordinatar
- Viss tiljengeleg: opningstider, kategori og telefonnummer
- Ein knapp som opnar staden i Apple Maps

I tillegg skal du implementere ein **kategori-spesifikk animasjon** øvst i visinga. Kvar kategori har sitt eige emoji-symbol og animasjonsstil:

## RESTAURANT

- Vis øvst i detaljvisinga
- Når visinga blir opna, skal emojien rotere 360°
- Animasjonen skal vare i 1 sekund
- Bruk `.easeInOut` som timings-funksjon
- 

## KAFÉ

- Vis ☕ øvst i detaljvisinga
- Tre "damp-partiklar" skal animerast oppover frå koppen
- Bruk `Text("")` eller `Circle()` for partiklar
- Kvar partikkel skal fade ut medan han bevegar seg oppover (opacity frå 1.0 til 0.0)
- Animasjonen skal gjentakast kontinuerleg
- Bruk `.easeOut` som timings-funksjon

## HOTELL

- Vis øvst i detaljvisinga
- Emojien skal “bounce” når visinga blir opna
- Bruk `.spring()` med bounce-effekt
- Start frå `scale = 0.5` og animer til `scale = 1.0`

## OPPGÅVE 4

I denne oppgåva skal du implementere ein vurderingsfunksjon som let brukaren gje tilbakemelding på stader dei har besøkt eller ønskjer å tilrå. Vurderingane skal lagrast lokalt med SwiftData og lagrast mellom appkjøringar. Dette gjev appen eit personleg preg og gjev brukaren høve til å byggje opp sin eigen historikk.

Frå detaljvisinga skal brukaren kunne gje ei vurdering frå **1 til 5 stjerner**. I staden for å vise fem separate stjerner, kan du bruke **éi stjerne som blir gradvis fylt** for å indikere vurderingsnivået. Fyllingsgraden skal spegle gjennomsnittet av alle vurderingar for staden:

- 1 stjerne = 20 % fylt
- 5 stjerner = 100 % fylt

Brukaren skal kunne gje fleire vurderingar til same stad over tid. Kvar vurdering skal lagrast som ei **eiga oppføring** i SwiftData.

Appen skal rekne ut **gjennomsnittleg vurdering** for kvar stad og vise dette som ei fylt stjerne. Rund av til nærmeste heile stjerne. Dersom ein stad ikkje har nokre vurderingar, skal det ikkje visast noka stjerne. Stjernene skal visast på kartet og i lista.

## OPPGÅVE 5

I denne oppgåva skal du utvikle ein fleksibel og brukarvenleg søkjefunksjon som let brukaren finne stader basert på tekst, kategori, radius og sortering. Målet er å gje brukaren full kontroll over kva stader som blir viste, og korleis dei blir presenterte – både i kart og liste.

Appen skal innehalde eit **søkjefelt** der brukaren kan skrive inn namn eller nøkkelord for å finne spesifikke stader. Søk skal skje i **real-time**, og resultata skal oppdaterast medan brukaren skriv. For å unngå overbelasting av API-et, må du implementere **debounce** med minimum 300 ms forseinking før søket blir trigga.

Brukaren skal også kunne justere **søkjeradius** med ein **Slider**, frå 1 til 10 km. Radiusen skal visast tydeleg og brukast som parameter i API-kallet. Standardverdien kan vere 5 km frå Oslo S (59.9111, 10.7503), eller frå det noverande sentrumet til kartet og skal lagrast ved hjelp av AppStorage.

Søket skal kunne kombinerast med **kategori-filteret** fra Oppgåve 2 (restaurantar, kafear, hotell). I tillegg skal brukaren kunne velje **sorteringsrekkefølgje**:

- Avstand (nærast først)
- Rating (høgast først)
- Alfabetisk (A–Å)

Brukaren skal også kunne velje om søket skal gjelde **alle stader**, eller berre **lagra favorittar**. Favorittar blir lagra med SwiftData og skal kunne filtrerast separat.

Det skal vere ein **clear-knapp** som nullstiller søkerfelt, radius, kategori og sortering, og rettar opp igjen standardvisning.

## ARBEIDSKRAV

Not all those who  
wander are lost.” – J.R.R.  
Tolkien

- Real-time søk med debounce (minimum 300 ms)
- Loading state (**ProgressView**) medan søker skjer
- Handter scenario der ingen resultat finst (t.d. vis “Ingen treff” med ikon eller emoji)
- Bruk SwiftData til å lagre og hente favorittar
- Implementer **pull-to-refresh** i listevisinga for å hente nye data manuelt

# OPPGÅVE 6

Spel inn eit maksimum 5-minutts skjermopptak der du går gjennom koden din. Her skal du vise koden i videoen medan du snakkar.

## ARBEIDSKRAV:

**API-integrasjon :** Vis korleis data flyt frå Geoapify API til UI. Forklar korleis du parsar JSON og handterer async/await.

**SwiftData:** Forklar korleis rating blir lagra og henta. Vis modellen og korleis du reknar ut gjennomsnitt.

**Animasjonar:** Vel éin animasjon og forklar linje for linje korleis han fungerer.

**Utfordringar:** Kva var vanskelegast? Korleis løyste du det?

Tips: Bruk QuickTime Player (File | New Screen Recording) eller Cmd+Shift+5 på Mac. Snakk medan du viser koden. Legg ved fila i mappa saman med koden.

