

**Emnekode:** TDS200

**Emnenavn:** Kryssplattform

**Vurderingskombinasjon:** Skriftlig individuell hjemmeksamen (A-F)

**Utlevering/innleveringsdato:**

**Filformat:** .zip

## KONTINUASJONSEKSAMEN




### TDS200 Kryssplattform

Tillatte hjelpemidler: Alle

Høsten 2022

---

Oppgavesettet består av to oppgaver hvorav begge skal besvares. Du vurderes på både teknisk og teoretisk kompetanse. Resten av oppgavesettet følger på de neste sidene. Strukturen på innleveringen skal være som følger:

 kandidatnummer.zip	
 Oppgave1.pdf	→ 1500 ord teoretisk besvarelse på case-oppgave
 Oppgave2-readme.txt	→ App-funksjonalitet og emulator/simulator du har testet på
 TDS200_h22_kont_kandidatnummer.zip	→ Mappe med app-kode

#### Viktig informasjon

- Følg alltid god referanseskikk. Dersom du tar i bruk sitater, kodesnutter, statistikk, bilder, figurer og lignende elementer du ikke selv har laget eller skrevet, skal dette refereres til for å skille mellom eget og andres arbeid. Referansene/kildehenvisningene gjøres etter [Høyskolen Kristianas gjeldende retningslinjer](#).
- Bruk av HTML/Vue-kode fra Ionic sitt [komponentbibliotek](#) er tillatt for å bygge grensesnitt.
- Plagiat skal ikke forekomme. Det er ikke tillatt å kopiere kode uten å referere til hvor koden er funnet. Det eneste unntaket til regelen er det overnevnte komponentbibliotek fra Ionic, hvor *HTML* -kode kan kopieres fra. Bruk av npm-moduler går ikke innunder plagiat, og det oppfordres til å vise kompetanse i bruk av npm/tredjepartsbiblioteker – husk referanser.

## Oppgave 1 – Teoretisk kompetanse (~35 poeng)

Rammen for denne oppgaven er maksimalt **1500 ord, ekskludert referanseliste**.

### Oppgavetekst

Felles for de to deloppgavene i denne oppgaven er at de skal besvares med egne ord. Eventuelle sitater skal refereres til for å skille mellom eget og andres arbeid. Suppler med kodesnutter, skjermbilder, figurer/modeller o.l. som kan hjelpe med å kommunisere svaret ditt. Besvarelsen leveres som PDF eller Word-fil. Poengene oppgitt er veiledende antall poeng for hver oppgave, og reflekterer forventet innsats. For høy måloppnåelse er det forventet en kombinasjon av egne erfaringer og tanker, samt eksterne kilder. Besvar følgende deloppgaver:

1. I TDS200 Kryssplattform har vi brukt Ionic og Capacitor for å lage hybrid-apper. Det finnes også alternative rammeverk, for eksempel React Native, NativeScript og Flutter. Ta for deg ett av de tre nevnte alternative rammeverkene, og beskriv hvordan det valgte rammeverket differensierer seg fra Ionic og Capacitor. (~20 poeng)
2. På hvilke måter skiller hybrid-apputvikling seg fra tradisjonell native-utvikling? Ta for deg de ulikhetene du mener er viktigst å ta høyde for som app-utvikler. (~15 poeng)

## Oppgave 2 – Teknisk kompetanse (~65 poeng)

For denne oppgaven er det kun kombinasjonen **Ionic + Vue + TypeScript/JavaScript + Capacitor** som skal brukes.

### Ved sensur vektlegges blant annet følgende punkter:

- Løsningen fungerer (bygger og kjører korrekt), og kan kjøres i både nettleser (ionic serve) og på Android-emulator og/eller iOS-simulator.
- TypeScript-funksjonalitet brukes der det gir mening for å gjøre kodebasen bedre, f.eks. spesifisering av returverdityper, variabler, parametere og i klasser.
- Forventet bruk av Vue-funksjonalitet som eksempelvis *binding*, *direktiver* (v-if, v-for, osv.) og komponenter der det gir kode-, struktur- og prosjektmessig mening.
- Kommenter koden for å illustrere forståelse for hva koden gjør (les: dette betyr ikke at *all* kode krever kommentering. Tenk at kommentarer skal vise til sensor at **du** forstår hva koden du har levert faktisk gjør).

### Oppretting og levering av eksamensprosjekt:

- Ved oppretting av eksamensprosjektet, ta utgangspunkt i følgende kommando:  
`ionic start TDS200_h22kont_kandidatnummer blank --type vue`

*Ikke bruk annen mal enn «blank» ved oppretting av prosjektet. Hvis du ønsker å ha faner eller sidemeny som en del av appen, må det implementeres selv, ikke genereres gjennom bruk av respektiv mal.*

- Legg ved en fil kalt Oppgave2-readme.txt hvor du beskriver hvilken funksjonalitet du har implementert i appen, og om du har kjørt den på iOS-simulator, Android-emulator eller begge.
- Ved levering av prosjektet skal **ikke** mappen `node_modules` inkluderes.
- Sørg for at filen `package.json` inneholder eventuelle eksterne npm-pakker du velger å ta i bruk i prosjektet ditt, forvent at sensor installerer npm-pakker med kommandoen `npm install`.

## Oppgavetekst teknisk oppgave: Offline søvndagbok

**Case:** En stadig økende andel av befolkningen opplever søvnproblemer ([Folkehelseinstituttet, 2021](#)). Én måte å kartlegge søvn på er en søvndagbok, for eksempel med en papir-løsning slik som den fra [Helse Bergen](#). Din oppgave er å lage en *offline* søvndagbok-app med Ionic. Hverken Directus eller annen backend-løsning skal brukes i dette prosjektet, all data skal oppbevares lokalt på telefonen (*les «[Data Storage in Capacitor](#)» for dokumentasjon på bruk av Capacitor Preferences API*).

### Minstekrav til implementasjon (ikke prioritert rekkefølge):

- Som bruker skal jeg kunne legge til nytt søvndagbok-innlegg. Det bør på det minste inneholde en tittel, klokkeslett for leggetid, og notis om opplevelsen av nattens (manglende) søvn. For eksempel på flere relevante felter, se søvndagboken til Helse Bergen (*lenket til over*). Alle søvndagbok-innleggene må lagres lokalt på telefonen, slik at de er tilgjengelige selv hvis appen stenges ned og startes på nytt.
- Presentere en oversikt over alle dagbok-innleggene som er tilgjengelig. Dette kan for eksempel være i liste- eller kalenderformat.
- Hvert innlegg må det gå an å trykke seg inn på for å få mer informasjon om den loggførte søvnen.

De tre punktene over utgjør minstekravet til oppgaven. For de som sikter høyt karaktermessig forventes mer funksjonalitet og kompleksitet i implementasjonen. For *eksempler* på funksjonalitet som kan være med på heving av karakter, se listen under. Husk også at sensor må kunne teste appen fra A til Å med minimalt av manuell konfigurasjon.

### Eksempler på funksjonalitet som kan heve karakter:

- Gi bruker mulighet til å sette et passord/pin-kode som sikkerhet, slik at ingen andre enn brukeren selv kan lese innleggene i dagboken eller legge til nye innlegg.
- Implementeres punktet over kan du også implementere router guards for å sørge for at ikke uvedkommende kan navigere til undersider de ikke skal ha tilgang til.
- GPS (*Capacitor Geolocation*) for å få med brukerens lokasjon ved oppretting av nytt innlegg, f.eks. for å på sikt kunne se om man sover bedre på visse steder.
- Tilbakemeldinger/feedback ved bruker-interaksjon (feilmeldinger, loading-spinnere, m.m.).
- Din egen vurdering ("*rating*") av nattens søvn (tenk *IMDB-rating*, «1-5 stjerner»).
- Lokale varslinger (*(scheduled) local notifications*) for å sørge for at brukeren husker på å fylle ut søvn-detalljer hver dag.
- Bruk av Google Maps eller annen kartleverandør for fremvisning av lokasjonen på en gitt natts søvn («*denne natten sov jeg godt, og da sov jeg visst på hytta*»).
- Bruk av npm-moduler (`npm install [pakkenavn] --save`).
- Bruk av relevante eksterne API-er (REST/GraphQL).
- Innslag av eget design.
- Implementasjon av accessibility-prinsipper.