Eksamen

DTS200 Kryssplattform - Høsten 2021

Generell informasjon

Eksamen er delt inn i 2 oppgaver som er knyttet til hverandre. Oppgave 1 er en praktisk programmeringsoppgave. Oppgave 2 er en skriftlig oppgave som omhandler oppgave 1.

- Dette er en individuell hjemmeeksamen.
- Det skal utvikles en app i React Native.
- Appen skal skrives og leveres i en Expo managed workflow (et standard Expo-prosjekt).
- Appen skal kjøre på iOS og Android.
    - Den kan inneholde operativsystem-spesifikke ulikheter, men den funksjonelle flyten skal være nogenlunde lik i begge implementasjonene.
- Eksterne ressurser som er beskyttet av copyright kan brukes. Dette er fordi denne kodebasen ikke skal publiseres offentlig.
- Tredjeparts NPM-biblioteker kan gjerne brukes dersom det er nødvendig eller fornuftig.
- Kreativitet applauderes
    - Hva kan du lage som er spennende?
    - Hvilke morsomme funksjoner kan du finne på?
    - Kan du visualisere dataen din på en engasjerende måte?
- Kontakt gjerne lærer i god tid før innleveringsfristen dersom du ønsker å diskutere løsningen din.

Lykke til!

Oppgave 1 - Praktisk oppgave

Du skal utvikle en app i React Native, ved bruk av Expo Managed Workflow for iOS og Android.

Oppgaven gir deg frihet til å bestemme selv hva applikasjonen din skal være, men skal omhandle et eller flere API-er.

Appen må gjerne implementere flere ulike native funksjonaliteter som operativsystemet tilbyr. Disse er som regel tilgjengelig via eksterne biblioteker.

I denne oppgaven skal det ikke lages noen backend. Dersom du ønsker å legge til rette for kommunikasjon mot en backend kan du simulere kall som om du normalt skulle kalt et API. Eksempelvis kan en simulert backend jobbe mot lokale .json-filer. Dette er ikke et krav, men kan åpne opp for flere funksjonelle muligheter.

Besvarelsen skal være en (1) .zip-fil som inneholder hele Expo-prosjektet, uten node_modules-mappen.

Når oppgaven unzippes skal sensor kjøre `<$ npm install>` etterfulgt av `<$ expo start>`.

Appen må dekke følgende funksjonaliteter:

1. Appen skal liste ut data. Her kan det brukes enten listekomponenter eller andre måter å visualisere lister på.
2. Søk: brukeren kan gi en input og systemet skal gjøre oppslag.
3. Detaljert visning av ressurser som er hentet fra et API.
4. Det skal brukes rammeverk for å navigere mellom ulike skjermbilder.

API-er

Du skal lage en app som omhandler et eller flere av de følgende API-ene.

Fiction

Rick And Morty API - https://rickandmortyapi.com/
Star Wars API - https://pipedream.com/apps/swapi

Vær

Yr - https://developer.yr.no/

Covid 19

Disease.sh - https://disease.sh/docs/#/
Johns Hopkins University - https://www.mongodb.com/developer/article/johns-hopkins-university-covid-19-rest-api/

Oppgave 2 - Skriftlig oppgave

Formålet med denne delen av oppgaven er at du skal demonstrere forståelse for koden du skriver. Du skal kunne formidle og reflektere om arkitektur-mønstre, veivalg, eller andre kompleksiteter i kodebasen din.

Dette er ingen lang skriveoppgave. Maks 900 ord, gjerne kortere. Kutt ned på unødvendig informasjon. Å skrive kort er en kunst.

Dokumentet skal leveres som en .pdf-fil og leveres som en del av samme .zip-fil som kodebasen. Dokumentet skal hete documentation.pdf og ligge på øverste mappe-nivå (root).

Dokumentet skal deles inn i følgende punkter:

Del 1: Introduksjon til appen
- Hva har du laget?
- Skriv kort om den funksjonelle flyten i appen.

Del 2: Refleksjoner
- Reflekter rundt arkitektoniske valg som er tatt. Dette kan f.eks være hvordan komponenter er skrevet generiske, eller hvordan du har lagt til rette for separation of concerns og single responsibility principle.
- Dersom du har hatt noen utfordringer underveis, hvordan løste du disse? Lærte du noe?
- Ville du gjort noe annerledes neste gang?
- Du må gjerne tegne diagrammer dersom du synes dette er nyttig.

Tips

Begynn med å sette deg inn i hvilke API(er) du ønsker å bruke.

- Hvordan bruker en dette API-et?
- Sett deg inn i datasettet. Hva er tilgjengelig?

Jobb frem en idè før du begynner å programmere.

- Hvilke funksjonaliteter ønsker du å jobbe med, og hvordan skal de fungere?
- Lag deg selv noen enkle wireframes og/eller skisser.
    - Hvor skal knapper stå?
    - Hvilke tekster skal du ha?
    - Hvilke funksjonaliteter skal eksistere, og hvor i flyten?
- Hvordan skal den tekniske implementasjonen henge sammen?

Bruk tid på det visuelle.

- Hvordan kan du presentere dataene og funksjonene dine på en måte som føles engasjerende for brukeren?

Bruk TypeScript

- TypeScript hjelper deg med og tvinger deg til å skrive god og gjennomtenkt kode.
- En typesiker kode er mer lesbar for andre.
- Du demonstrerer flere evner, og viser forståelse for kompleksitet.

Sensur

Alle prosenter er estimater.

Oppgave 1 (≈80%)	
Kodestruktur (≈30%)	Eleven viser forståelse for - Ryddig kode, gode navn, struktur - Gjenbruk, custom hooks, generiske komponenter - Single responsibility principle - Typesikkerhet ved bruk av TypeScript
API-implementasjon (≈20%)	Det skal være lagt opp til - Adskilt kompleksitet fra resten av applikasjonen - Gjenbruk - Single responsibility principle
Grensesnitt (≈30%)	Applikasjonen er intuitiv og appellerende å bruke. Det er lagt vekt på design og presentasjon. Flyten og bruksmønstre i applikasjonen skal gi mening og følge standarder som finnes i industrien og på operativsystemet. Grensesnittet er like fungerende på tvers av operativsystemer og skjermstørrelser.

Oppgave 2 (≈20%)	
Skriftlig oppgave	Eleven skal - vise forståelse for arbeidet som er utført. - evne å formidle komplekse temaer som f.eks arkitektur, generiske komponenter og gjenbruk av kode. - reflektere rundt sitt eget arbeid.

I tillegg vil også følgende punkter påvirke den totale vurderingen:

- Applikasjonens omfang
  - En stor og kompleks applikasjon vil veie tyngre enn en liten applikasjon med få funksjoner.
- Kompleksitet
  - Bruk av avanserte og sofistikerte teknikker og mønstre.
- Kodestruktur og vanskelighetsgrad
  - Generelt ryddig og lesbar kode og struktur.
  - Evne til å lage komplekse implementasjoner.