Krav til produktet

Det er mulig å oppnå totalt 35 poeng på produktet, koden, modelleringen og beskrivelsen rundt arkitekturen. Poeng blir gitt basert på følgende kriterier:

Generelle krav

- Det som redegjøres for i rapporten bør reflekteres i appen (spesielt funksjonelle og ikke-funksjonelle krav).
- Produktet som utvikles skal være en besvarelse på caset dere har valgt.
- De obligatoriske API-ene for caset dere har valgt må anvendes. Dere kan supplere med andre API-er i tillegg.
- Prosjektet må ligge i Github-repoet som ble opprettet for ditt team (på github.uio.no) og git-loggen skal inkluderes.
- Alle prosjekter **må** inneholde minimum 3 markdown-filer:
    - README.md:
        - Beskriver hvor dokumentasjonen ligger og hvordan man kjører appen. Denne burde også inneholde informasjon om biblioteker som er brukt.
    - ARCHITECTURE.md:
        - Beskriver arkitekturen som er benyttet i appen.
        - Beskrivelse av hvordan viktige objektorienterte prinsipper som lav kobling og høy kohesjon samt design patterns som MVVM og UDF er ivaretatt i løsningen burde også være med.
        - Beskriv løsningen beregnet på lesere som skal jobbe med drift, vedlikehold og videreutvikling av løsningen. Beskriv hvilke teknologier og arkitektur som brukes i løsningen. Beskriv hvilket API-nivå (Android versjon) dere har valgt, og hvorfor.
    - MODELING.md:
        - Beskrivelse og diagrammer, vi anbefaler å generere dem med Mermaid som vist på forelesning. Se kravene til modellering lenger ned.
- Kotlin skal primært benyttes som programmeringsspråk og Jetpack Compose skal primært benyttes til å lage brukergrensesnitt.

Tekniske krav

Arkitektur og kodeskikk

- Koden skal være enten på engelsk eller norsk.
- Appens filer bør være logisk inndelt i en mappestruktur med lagdeling.
- Koden bør være ryddig, lesbar, godt dokumentert og kommentert. Vi anbefaler å bruke automatisk formatering.
- Koden skal følge etablerte design patterns (f.eks. MVVM, UDF) og de sterkt anbefalte praksisene for Android-utvikling.

Robusthet

- Appen skal ikke krasje på valgt API-level (dere velger API-level selv ved opprettelse av prosjektet. Redegjør for valg av API-level. Den skal heller ikke krasje ved feil som at internett mangler eller at man ikke får hentet noe fra API, men heller varsle om dette.
- Antall «Warnings» og «Errors» fra IDE-en bør være så lavt som mulig. Redegjør for eventuelle «Warnings».
- Appen bør støtte Activity Lifecycle, og sørge for at viktig informasjon ikke forsvinner dersom appen blir avsluttet ved et avbrudd, enheten blir rotert eller brukeren navigerer til andre skjermer.
- Kall mot API-er skal gjøres pålitelig (dvs. at kall gjøres asynkront og IFIs proxyserver skal benyttes mot MET sine tjenester).

Testing

- Det skal være skrevet automatiske enhetstester (minst 10) til noe av funksjonaliteten i appen.

Tilgjengelighet

- Appens layout bør være adaptiv eller responsiv slik at appen fungerer godt uavhengig av skjermstørrelse og skjermoppløsning.
- Appen bør ta hensyn til kravene til universell utforming (WCAG 2.1-standarden) og bruke anbefalte tilgjengelighetsverktøy i Jetpack Compose.

Modellering og systemdesign

- De viktigste funksjonelle kravene til applikasjonen bør beskrives – bruk gjerne use case diagram, samt sekvensdiagram og tekstlig beskrivelse av de viktigste use-casene.
- Modelleringen bør også inneholde klassediagram som reflekterer use-case og sekvensdiagrammene.
- Andre diagrammer bør også være inkludert for å få frem andre perspektiver, for eksempel aktivitetsdiagram (flytdiagram) eller tilstandsdiagram.