2023

AvtaleApp

FAG: DAVE3600 / MAPPEINNLEVERING 2

SULTAN AVTAJEV / S199219

Innhold

Innledning 2
Oppgaven 3
Spesifikasjoner og krav 4
Generelt 4
Brukergrensesnitt 4
Datahåndtering 4
Notifikasjoner og meldinger 4
Bakgrunnstjenester 4
Funksjonelle krav 4
Ikke-funksjonelle krav 4
Teknologiske valg 5
Utviklingsmiljø 5
Datalagring 5
Arkitektur og kode 5
Notifikasjoner og meldinger 5
Bakgrunnstjenester 5
Navigasjon 5
Design og brukervennlighet 6
Designprinsipper 6
Fargevalg og ikoner 6
Utfordringer og løsninger 6
Ingen forhåndsdesign 6
SMS-tillatelser og sikkerhet 6
Fremdriftsplan 7
Screenshots 8
Testing 9
Enhetstesting 9
Manuell Testing 9
Brukertester 9
Konklusjon 9
Fremtidige forbedringer og utvidelser 9
Kilder 10

Innledning

Hei, og velkommen til denne dyptgående rapporten! Hvis du har interesse for hvordan teknologi kan effektivisere og automatisere din daglige planlegging og forvaltning av avtaler, da er du på rett sted. Rapporten vil fokusere på en innovativ app jeg har utviklet, kalt "AvtaleApp". Denne appen er designet for å hjelpe brukere med å holde styr på sine avtaler, samtidig som den gir muligheten til å sende automatiske påminnelser til venner, familie eller kollegaer.

Appen er utformet for å være både intuitiv og kraftig, noe som gjør den ideell for et bredt spekter av brukere; fra de som bare ønsker en enkel måte å spore viktige datoer, til de som trenger en fullverdig løsning for å forvalte en kompleks kalender. Den er spesielt nyttig for profesjonelle, studenter og faktisk alle som har et hektisk liv og trenger å effektivisere sin tid.

I denne rapporten vil jeg gi deg en grundig gjennomgang av hvordan jeg utviklet "AvtaleApp", fra den tekniske arkitekturen til brukergrensesnitt-design. Jeg vil også inkludere innsikter i hvordan jeg forsøkte å gjøre appen så brukervennlig og fleksibel som mulig. Så, forbered deg på en lærerik reise gjennom "AvtaleApp"!

Oppgaven

Denne appen skal holde rede på avtalene dine. Det skal kunne registreres venner med navn og telefon og det skal registreres avtaler med dato, klokkeslett og treffsted. Alt skal lagres i en SQLite database på telefonen.

Når man kommer inn i appen skal det vises en liste over avtaler. Man skal kunne komme til skjermbilder som viser en liste over alle venner som er registrert og til preferanser.
Det appen skal benyttes til er at den skal sjekke databasen en gang i døgnet. Er det en avtale skal det sendes en sms til vennene det gjelder og det skal komme opp en notifikasjon på telefonen.

I preferanser som skal implementeres med en xml-fil og preferansefragment skal man kunne slå sms-tjenesten av og på. Default skal tjenesten være av. Det skal kunne velges når på døgnet sms skal sendes. Default skal dette være 06.00. Det skal også være mulig å sette default melding som skal sendes i smsen.

Avtaler og venner skal kunne legges til og slettes og evt. endres. Når tjenesten slås på skal det sendes et egendefinert broadcast til en BroadcastReceiver MinBroadcastReceiver som skal kalle på en service MinPeriodisk som setter AlarmManager til å kjøre en service MinSendService som sjekker databasen og sender ut eventuelle påminnelser til tidspunktet satt i SharedPreferences.
Positivt hvis dere lager en ContentProvider som gjør det mulig å dele vennedata med andre applikasjoner.

Dere velger selv om en avtale kan være med en eller flere venner. Dere velger selv om dere vil benytte aktiviteter eller fragmenter. Dere velger selv om dere vil jobbe mot SQLite eller benytte ROOM. Dere velger selv om dere vil benytte ListView eller RecyclerView. Dere velger selv hvordan man navigerer mellom Avtaler, Venner og Preferanser.
Programmer mot API28 som sist. Lag rapport som sist og navngi prosjektet MED studentnr!
Høyreklikk på domenenavnet, velg Refactor, Rename. Ved uklarheter ta kontakt.

Spesifikasjoner og krav

Generelt

- Appen skal kunne kjøre på Android-enheter med API 28 eller høyere.

Brukergrensesnitt

Hovedskjerm: Viser en liste over kommende avtaler.
Venneliste: Viser en liste over registrerte venner.
Preferanseskjerm: Lar brukeren endre appinnstillinger.

Datahåndtering

SQLite Database/ROOM: Lagrer all venn- og avtaleinformasjon.
SharedPreferences: Lagrer brukerpreferanser inkludert tidspunkt for SMS og frekvens for databasensjekk.

Notifikasjoner og meldinger

SMS: Sender SMS-påminnelser til venner om kommende avtaler.
- Standard tidspunkt: kl. 06:00
- Frekvens: En gang per dag på dagen av avtalen.

Lokale notifikasjoner: Viser notifikasjoner på enheten.
- Standard tidspunkt: kl. 06:00
- Frekvens: En gang per dag på dagen av avtalen.

Bakgrunnstjenester

Periodisk sjekk: En service som kjører periodisk for å sjekke for kommende avtaler og sender SMS/notifikasjoner.
Frekvens: En gang per dag
Tidspunkt: Samtidig som SMS og notifikasjoner, eller konfigurerbart i preferanser.

Ekstra

ContentProvider: For deling av vennedata med andre applikasjoner (valgfritt).

Funksjonelle krav

- Brukeren skal kunne legge til, redigere og slette venner og avtaler.
- Appen skal sende SMS-påminnelser og vise lokale notifikasjoner for kommende avtaler.
- Brukeren skal kunne endre preferanser for når og hvordan påminnelser sendes.
- Brukeren skal kunne endre preferanser for språk mellom norsk og engelsk
- Appen skal ha en app-logo på skrivebord

Ikke-funksjonelle krav

Responsivt design: Appen skal være brukervennlig på forskjellige skjermstørrelser.
Ytelse: Appen skal kjøre jevnt og effektivt.
Sikkerhet: Data skal lagres på en sikker måte, og SMS-tillatelser skal håndteres forsvarlig.

Teknologiske valg

Utviklingsmiljø

For utviklingen av denne appen er Android Studio benyttet, med Kotlin som programmeringsspråk. Kotlin er et moderne, statisk-typet programmeringsspråk som gir utviklere større fleksibilitet og effektivitet, spesielt for Android-utvikling. Det gir også forbedret sikkerhet og reduserer mengden boilerplate-kode sammenlignet med Java.

Datalagring

Data i appen er lagret på to måter: gjennom SQLite/ROOM for venn- og avtaleinformasjon, og SharedPreferences for brukerpreferanser. ROOM er en abstraksjon lagt på toppen av SQLite og gir objektrelasjonsmapping som forenkler databehandling. SharedPreferences brukes for å lagre brukerpreferanser som tidspunkt for SMS, frekvens for databasensjekk, og språkinnstillinger. Dette sørger for en smidig og personlig brukeropplevelse.

Arkitektur og kode

Koden er organisert i henhold til Model-View-ViewModel (MVVM) arkitekturen, som forenkler testing og vedlikehold. Her er noen av de sentrale klassene:

- **MainActivity**: Dette er inngangspunktet for appen og fungerer som vert for de ulike fragmentene.
- **AppointmentViewModel** og **FriendViewModel**: Disse ViewModel-klassene håndterer forretningslogikken og interagerer med databasen.
- **MultiStepDialogFragment**: Håndterer logikken for å legge til eller redigere avtaler.
- **SettingsActivity**: Lar brukeren endre ulike appinnstillinger.
- **FriendsActivity**: Viser en liste over registrerte venner og tillater CRUD-operasjoner.

Notifikasjoner og meldinger

Appen bruker Androids innebygde NotificationManager for lokale notifikasjoner og SmsManager for å sende SMS. Dette gir en naturlig og integrert opplevelse for brukeren.

Bakgrunnstjenester

For periodiske sjekker av kommende avtaler er en bakgrunnstjeneste implementert. Denne tjenesten kjører en gang per dag og utløser SMS-påminnelser og lokale notifikasjoner basert på brukerens preferanser.

Navigasjon

Navigasjonen i appen er enkel og intuitiv. Appen starter med MainActivity, der brukerne kan navigere til ulike deler som "Venneliste", "Kommende avtaler", og "Innstillinger". Navigasjonen er designet for å unngå unødvendig bruk av systemressurser, spesielt med tanke på aktivitetshåndtering og tilbakestilling av stacken.

Ekstra (valgfritt)

Appen inkluderer også en ContentProvider for potensiell deling av vennedata med tredjepartsapplikasjoner, selv om dette for øyeblikket er valgfritt og ikke fullt implementert.

Denne arkitekturen og teknologivalget gir en robust, skalerbar, og brukervennlig app som er enkel å vedlikeholde og utvide i fremtiden.

Design og brukervennlighet

Designprinsipper

Appen er designet med et klart fokus på brukervennlighet og funksjonalitet. Den følger Material Design-prinsippene fra Android for å sikre en konsistent og intuitiv brukeropplevelse. Appen er utviklet med stående modus som den primære orienteringen for å optimalisere skjermbruken.

Fargevalg og ikoner

Appen bruker en enkel og nøytral fargepalett for å appellere til et bredt publikum. Denne fargepaletten komplimenterer appens funksjonalitet og gjør det lett for brukeren å fokusere på innholdet.

Tilbakemelding til bruker

Appen gir umiddelbar tilbakemelding gjennom toast-meldinger og dialogbokser ved viktige handlinger som å legge til en ny avtale eller venn. Dette bidrar til en mer intuitiv og responsiv brukeropplevelse.

Utfordringer og løsninger

Ingen forhåndsdesign

I motsetning til tradisjonelle tilnærminger, der flytdiagrammer eller mockups ofte blir produsert i forkant, startet denne utviklingsprosessen direkte i Android Studio. Dette skyldes tidligere erfaring med plattformen og en klar visjon om appens formål og funksjonalitet. En fremdriftsplan ble laget for å sikre at alle funksjonelle og ikke-funksjonelle krav ble møtt.

SMS-tillatelser og sikkerhet

Et annet teknisk dilemma var håndteringen av SMS-tillatelser, som kreves for noen av appens funksjoner. Dette ble adressert ved å implementere en omfattende tillatelsessjekk ved oppstart, og ved å informere brukeren tydelig om hvorfor denne tillatelsen er nødvendig.

Tid og kompatibilitet

Appen skulle være kompatibel med Android API 28 eller høyere, noe som satte visse begrensninger på funksjonalitet. Dette ble tatt hensyn til i alle faser av utviklingen, og nødvendige tilpasninger ble gjort for å sikre kompatibilitet.

Denne tilnærmingen har vært lærerik og har gitt innsikt i hvordan en effektiv utviklingsprosess kan se ut når man allerede har en solid forståelse av målsettingene og teknologien involvert.

Fremdriftsplan

Del 1: Prosjektoppsett og database
- Nytt Android Studio-prosjekt (OK)
- Velg Java som programmeringsspråk (OK)
- Mål API 28 eller høyere (OK)

Legg til avhengigheter
- ROOM for databasen (OK)
- RecyclerView for lister (OK)

Opprett databasemodeller
- Friend og Appointment entiteter (OK)

Sett opp ROOM Database
- Opprett DAO (Data Access Object) for både Friend og Appointment (OK)

Del 2: Brukergrensesnitt og datahåndtering
- Lag hovedaktivitet med RecyclerView
- Dette skal vise en liste over avtaler

Koble RecyclerView med databasen
- Vis avtaler fra databasen i listen

Implementer funksjonalitet for å legge til og slette avtaler
- Bruk ROOM for databasemanipulasjon

Opprett og implementer Preferanseskjerm
- Bruk SharedPreferences

Del 3: Notifikasjoner, SMS og Testing
- Implementer SMS-funksjonalitet
- Bruk Android's SmsManager

Implementer lokale notifikasjoner
- Bruk NotificationCompat

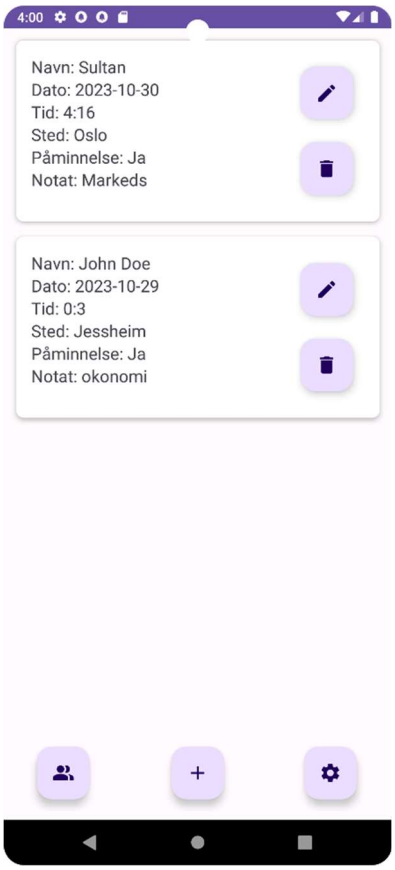
Sett opp bakgrunnstjeneste for periodiske sjekker
- Du kan bruke AlarmManager eller WorkManager

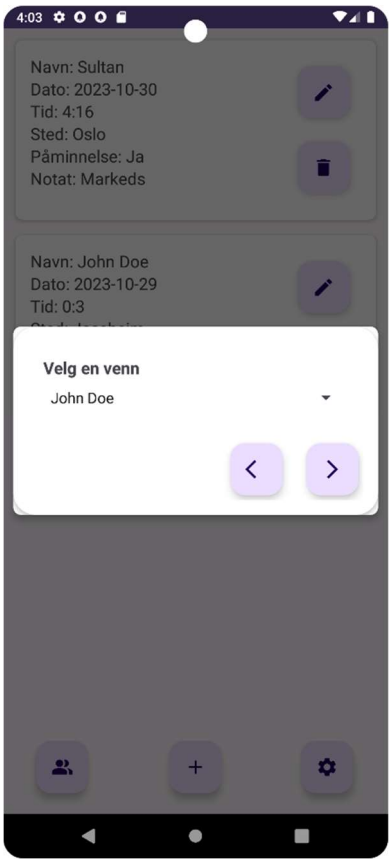
Test alle funksjonaliteter
- Sjekk at alt fungerer som forventet

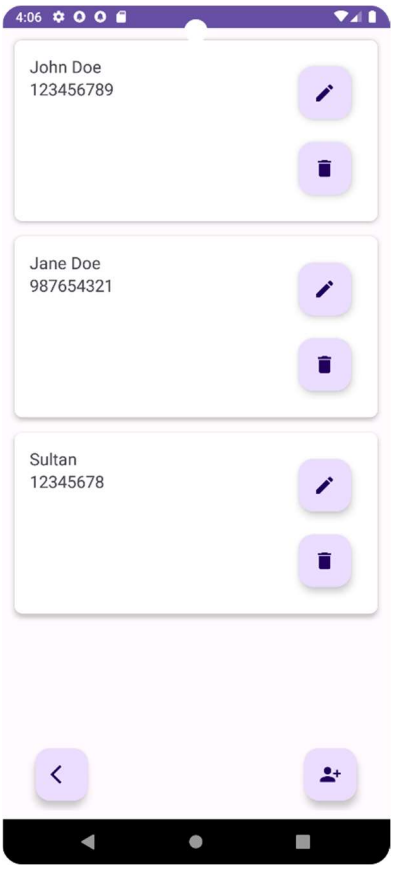
Del 4: Sluttfasen

Utfør en sluttrunde med testing
- Sørg for at alt er som det skal
- Legg til nødvendig dokumentasjon og kommentarer
- Forbered for innlevering eller presentasjon
- Gjør alt klart for å levere prosjektet

Screenshots







Testing

Testing har vært et nøkkelaspekt i utviklingsprosessen for å sikre at appen fungerer som den skal og tilbyr en god brukeropplevelse. Her er noen av metodene jeg har benyttet:

Enhetstesting

JUnit har blitt brukt for å utføre enhetstester på viktige deler av appen, inkludert datalagring i SQLite databasen og ROOM, samt logikken bak SMS-påminnelser og lokale notifikasjoner.

Manuell Testing

Manuell testing har også vært utført for å sjekke brukerinteraksjoner og UI-elementer. Dette inkluderer testing av hvordan appen oppfører seg på forskjellige Android-enheter og API-nivåer.

Brukertester

Gitt tidsrammen var det ikke mulig å gjennomføre en fullskala brukertest, men feedback fra kolleger og venner har blitt brukt til å identifisere og rette opp i mindre brukergrensesnittproblemer.

Konklusjon

Dette prosjektet har vært en innsiktsfull og lærerik erfaring i utviklingen av en app for avtalestyring og påminnelser. Selv om det var noen tekniske utfordringer knyttet til datalagring og tillatelseskontroll, har appen lykkes i å tilby en robust og funksjonell løsning for sitt hovedformål. Det har også gitt rom for videre forbedringer og utvidelser.

Fremtidige forbedringer og utvidelser

Appen har et solid fundament, men det er flere områder der den kan utvikles videre:

- **Integrering med andre kalenderapplikasjoner:** Dette vil tillate brukere å ha en sentralisert visning av alle sine avtaler.
- **Utvidet påminnelsessystem:** Legge til flere kanaler for påminnelser, for eksempel e-post eller push-varsler.
- **Avansert sortering og filtrering:** Forbedre brukeropplevelsen ved å tillate avanserte sortering og filtrering av avtaler og venner.
- **Forbedret sikkerhet:** Implementere ytterligere sikkerhetslag, for eksempel to-faktor autentisering.
- **Omfattende brukertesting:** Gjennomføre mer grundig brukertesting for å samle mer detaljert tilbakemelding.
- **Brukergrensesnittforbedringer:** Kontinuerlig oppdatering av designet for å forbedre brukeropplevelsen.
- **Flerspråklig støtte:** Til tross for at dette ikke var en del av de opprinnelige kravene, vil flerspråklig støtte være en viktig funksjon for en mer inkluderende app.
- **Logo og branding:** Introduksjon av en app-logo og ytterligere branding kan bidra til en mer profesjonell og tiltalende brukeropplevelse.

Kilder

https://developer.android.com/develop/ui/views/layout/declaring-layout¨
https://developer.android.com/develop/ui/views/layout/linear
https://developer.android.com/develop/ui/views/layout/relative
https://developer.android.com/develop/ui/views/layout/binding
+ Video og annet materiale tilhørende faget på canvas
+ Brukt ChatGPT for å forstå konseptene og hvordan de skal implementeres